THESIS

A SYSTEMS ANALYSIS OF STRIKE NAVAL AVIATION TRAINING

by

Tyler Y. Nekomoto

June 2013

Thesis Advisor:	Matthew Boensel
Co-Advisor:	Ronald Giachetti

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A SYSTEMS ANALYSIS OF STRIKE NAVAL AVIATION TRAINING			**5. FUNDING NUMBERS**
6. AUTHOR(S) Tyler Y. Nekomoto			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The Commander Naval Air Forces is analyzing the entire Naval Aviation training process to deliver the same quality of training but at a lower cost. This thesis documents the Systems Engineering process completed to conceive, design, and develop a desktop model that supports the investigation of training alternatives. The Naval Aviation Proficiency Analysis model incorporates the General Aviator Learning Equation methodology developed by the Naval Air Warfare Center Training Systems Divisions' Human Performance Analysis and Instructional Systems Division into a user-friendly self-automated spreadsheet model. It analyzes downloading efforts—moving blocks of flights or simulators from a phase with higher platform operational cost to one with lower platform operational cost—and highlights the effects that downloading has on cost, hour, and skill proficiency differences across all pipeline skills. The next steps are to incorporate offloading (flights to simulators) and find optimal training solutions by incorporating an after-market solver.

14. SUBJECT TERMS Naval Aviation, Model, Strike Fighter Naval Aviation Training, Aviation Training, Aviation Pipeline, Pilot Training, Naval Flight Officer Training			**15. NUMBER OF PAGES** 149
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

A SYSTEMS ANALYSIS OF STRIKE NAVAL AVIATION TRAINING

Tyler Y. Nekomoto
Commander, United States Navy
B.S., University of Colorado, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2013

Author:	Tyler Y. Nekomoto
Approved by:	Matthew Boensel Thesis Advisor
	Ronald Giachetti Thesis Co-Advisor
	Cliff Whitcomb Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Commander Naval Air Forces is analyzing the entire Naval Aviation training process to deliver the same quality of training but at a lower cost. This thesis documents the Systems Engineering process completed to conceive, design, and develop a desktop model that supports the investigation of training alternatives. The Naval Aviation Proficiency Analysis model incorporates the General Aviator Learning Equation methodology developed by the Naval Air Warfare Center Training Systems Divisions' Human Performance Analysis and Instructional Systems Division into a user-friendly self-automated spreadsheet model. It analyzes downloading efforts—moving blocks of flights or simulators from a phase with higher platform operational cost to one with lower platform operational cost—and highlights the effects that downloading has on cost, hour, and skill proficiency differences across all pipeline skills. The next steps are to incorporate offloading (flights to simulators) and find optimal training solutions by incorporating an after-market solver.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BI	Basic Instruments Flight Stage
CNAF	Commander Naval Air Forces
CNATRA	Commander Naval Aviation Training
FAM	Familiarization Flight Stage
FORM	Formation Flight Stage
FRS	Fleet Replacement Squadron
GALE	General Aviation Learning Equation
HP/ISD	Human Performance Analysis and Instructional Systems Division of NAWCTSD
INCOSE	International Council on Systems Engineering
KSAs	Knowledge, Skills, and Abilities
NAE	Naval Aviation Enterprise
NAWCTSD	Naval Air Warfare Command Training Systems Division
PA	Precision Aerobatics Flight Stage
PMA-273	Naval Undergraduate Flight Training Systems Program Office
RI	Radio Instruments Flight Stage
SE	Systems Engineering
SNA	Student Naval Aviator
SNFO	Student Naval Flight Officer
TRACOM	Training Command
TRAWING	Training Air Wing
T&R	Training and Readiness

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

In the present constrained financial environment, Department of Defense is seeking areas to implement financial efficiencies. While searching for cost-reduction opportunities, the Commander of Naval Air Forces initiated an end-to-end analysis of Naval Aviation training. To support this, the Human Performance Analysis and Instructional Systems Division of Naval Air Warfare Center Training Systems Division (NAWCTSD) requested a desktop model capable of analyzing potential syllabus changes for the cost reduction effort. The model must compare baseline training syllabi against altered ones to highlight the difference in skill proficiency and cost savings from those alterations.

The Naval Aviation Proficiency Analysis (NAPA) model was conceived, designed, and implemented in Microsoft Excel, using Visual Basic for Applications, in response to this need. The systems approach taken to complete this effort followed a modified waterfall process to guide development of the model. Systems Engineering best practices produced a model capable of satisfying user needs, while providing required system behaviors needed to facilitate Naval Aviation training analysis.

NAPA is suited to run on any computer with Microsoft Excel (2007 or later) installed. Its extensible architecture supports all training pipelines, and it has been designed to support alternative analysis efforts in the future. NAPA provides the two capabilities required by the primary stakeholder: (1) to determine the overall effect that hour reductions or training alterations have on the overall proficiency of the graduating aircrew; and (2) to identify training combinations that satisfy required proficiency levels at the desired cost. The NAPA model is currently in use by NAWCTSD HP/ISD.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my loving wife, Mina: Your unconditional love and support throughout this process made all the difference. Without you, I would not be where I am today. To my sons, Tyson and TJ: Thank you for giving up your time with Daddy to allow me to finish.

I would also like to thank my close friend and squadron-mate LCDR Bill “R2” Evans for all your help strengthening the model code: Without your help, the model would run slower, be less effective, and be extremely fragile. Thanks for all of your assistance.

To Dr. Jennifer Fowlkes and Dr. Joseph Sheehan from the Human Performance Analysis and Instructional Systems Division at NAWCTSD: Thank you for your patience, time, and dedication to finding the best way to train naval aviators. You are great Americans.

Special thanks to Professor Matt Boensel and Professor Ron Giachetti for guiding my thesis effort.   I appreciate the candid feedback and insightful conversations while working on this project.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THESIS SCOPE

Today's poor financial environment stresses every governmental program. As budgets incorporate drastic reductions, analysts scramble to provide alternatives to minimize degradation to our war-fighting capability.

While searching for areas capable of supporting cost reductions, the Commander of Naval Air Forces (CNAF) initiated a "Street-to-Fleet" analysis of Naval Aviation training. The Human Performance Analysis and Instructional Systems Division (HP/ISD) of the Naval Air Warfare Center Training Systems Division (NAWCTSD) has been supporting this analysis from inception and needs a desktop model to support their analysis efforts. NAWCTSD HP/ISD started with the Strike training pipeline with the intent of applying the same methodology to other training pipelines in the future. NAWCTSD HP/ISD has compiled data for this pipeline and identified a desired methodology for analysis, but has not incorporated them into an automated, easy-to-use model, capable of quickly analyzing scenarios that decision-makers want to investigate.

This thesis documents the Systems Engineering (SE) process completed to conceive, design, and develop a desktop model capable of supporting the investigation of variations in the way the Navy currently conducts aviation training. Before describing specifics of the effort, some background information will help place this effort in context.

B. BACKGROUND

1. Organizational Structure of Naval Aviation Training

Naval Aviator and Naval Flight Officer (NFO) training and production are currently conducted by seventeen Training Command (TRACOM) squadrons, aligned under five Training Air Wings (TRAWINGs) located at five Naval Air Stations in the southeastern United States (Chief of Naval Air Training 2012). Although the ultimate responsibility to provide trained aircrew to the Fleet rests with Commander, Naval Air Forces (CNAF), it is accomplished through a relationship with the Chief of Naval Air Training (CNATRA), the TRACOM, and the various platform Fleet Replacement Squadrons (FRSs) seen in Figure 1.

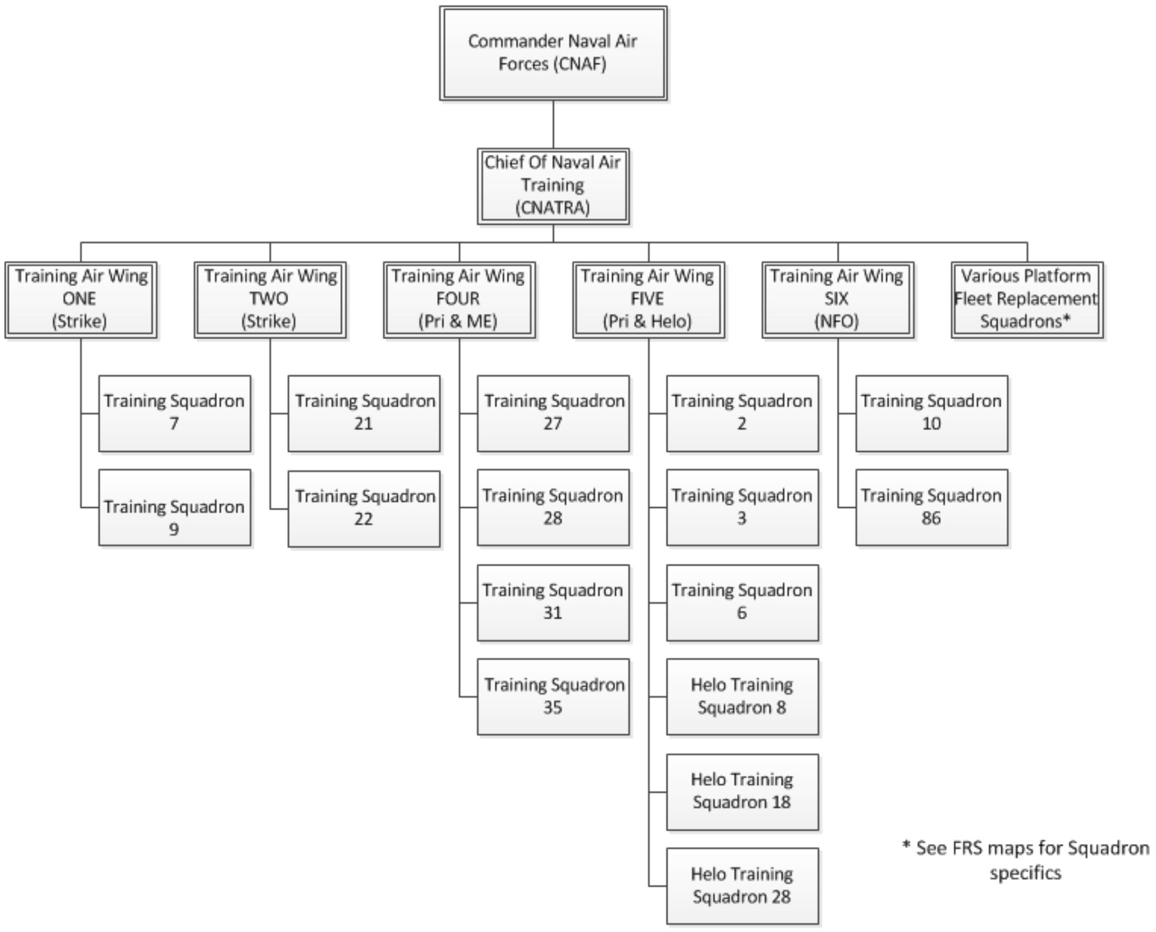

Commander Naval Air Forces (CNAF)
Chief Of Naval Air Training (CNATRA)
Training Air Wing ONE (Strike)
Training Air Wing TWO (Strike)
Training Air Wing FOUR (Pri & ME)
Training Air Wing FIVE (Pri & Helo)
Training Air Wing SIX (NFO)
Various Platform Fleet Replacement Squadrons*
Training Squadron 7
Training Squadron 9
Training Squadron 21
Training Squadron 22
Training Squadron 27
Training Squadron 28
Training Squadron 31
Training Squadron 35
Training Squadron 2
Training Squadron 3
Training Squadron 6
Helo Training Squadron 8
Helo Training Squadron 18
Helo Training Squadron 28
Training Squadron 10
Training Squadron 86
* See FRS maps for Squadron specifics

Figure 1. Organizational Chart of Naval Aviation Training

CNATRA is aligned under Commander, Naval Air Force, Pacific Fleet and CNAF–in the Naval Aviation Enterprise (NAE) concept. CNATRA also serves as the CNAF Deputy Commander for training, granting CNATRA the authority to orchestrate and manage the full spectrum of the training continuum–from student induction through the completion of FRS training (Chief of Naval Air Training 2012). FRS training becomes platform dependent and is provided at locations shown in Figure 2.

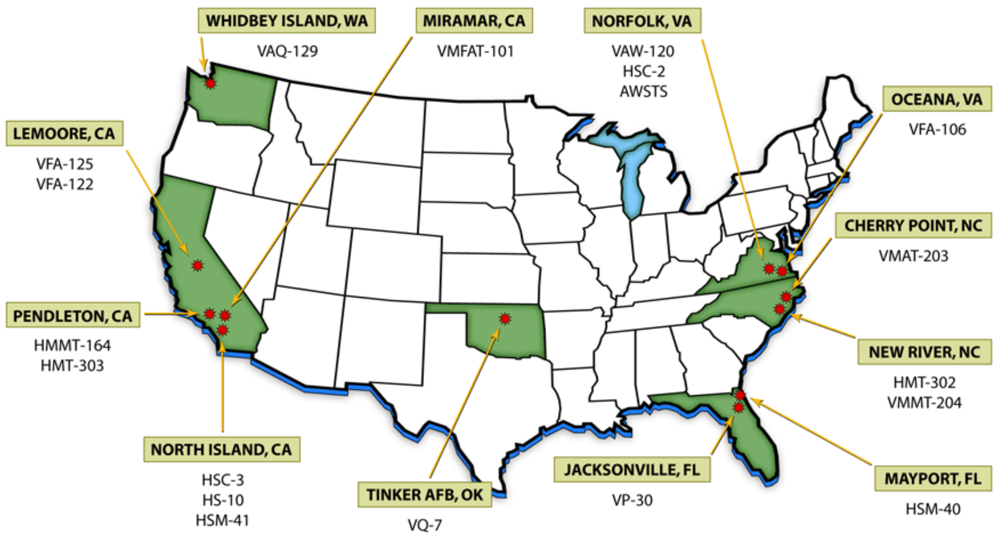


Figure 2. Current Fleet Replacement Squadrons and Locations for Naval Aviators (From Chief of Naval Air Training 2012)

According to CNATRA, having the ability to manage the entire continuum

> enables leadership and training practitioners to design and optimize training content and flow across all phases and pipelines of the entire training spectrum to ensure the right training is conducted at the right level. Additionally, this continuum and alignment ensures the most effective and efficient training organization is in place to achieve optimal student aviator time to train. This continuum and alignment ultimately ensures the production of the world's finest Aerial Warriors for the world's finest Air Force – the Naval Air Force. A Naval Air Force made up of Naval Aviators and NFOs, who can think, perform, Excel under pressure, and deliver in the most demanding aviation environment—projecting power ashore or at sea from the decks of aircraft carriers both day and night.(2012)

2. Big Business/Big Budget

CNATRA's annual flight budget for training is in excess of $575 million. A little more than 1,300 instructors fly approximately 750 TRACOM aircraft over 350,000 flight hours in order to provide slightly more than 1,500 pilots and NFOs to the Fleet each year. CNATRA operates almost a third of the Navy's aircraft inventory and accounts for nearly a third of its annual flight hours (Chief of Naval Air Training 2012).

CNATRA fulfills the needs of the Fleet through various training pipelines. The NFO pipeline / aircraft selection occurs throughout the continuum as shown in Figure 3

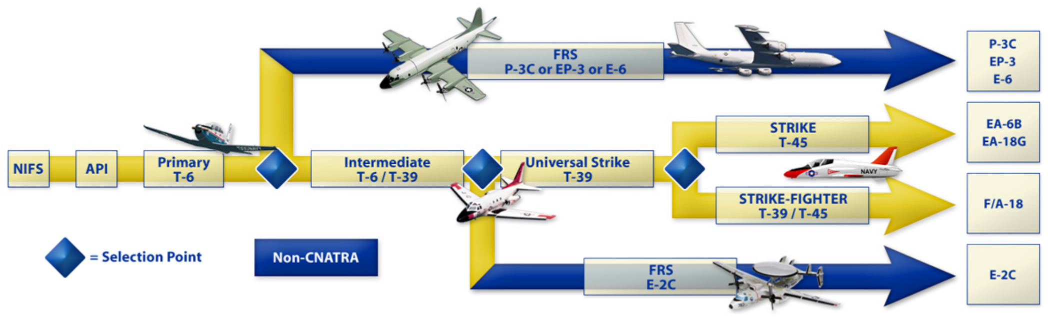


Figure 3. NFO Training Pipeline (From Chief of Naval Air Training 2012)

After Primary, SNFOs selected to fly as a navigator in a large, multi-engine aircraft, report directly to their selected platform FRS while the rest of the SNFOs continue on to the tactical (Intermediate) phase of training. Upon completion of Intermediate, SNFOs selected for E-2Cs report directly to the FRS, while Strike and Strike/Fighter SNFOs continue on to advanced training in the T-45C Goshawk.

Student Naval Aviators (SNAs) are selected for Maritime (multi-engine prop), E-2/C-2, Rotary (helicopters), Strike (jets), and the E-6 TACAMO after Primary flight training. Follow-on platforms are pipeline specific and can be seen in Figure 4.

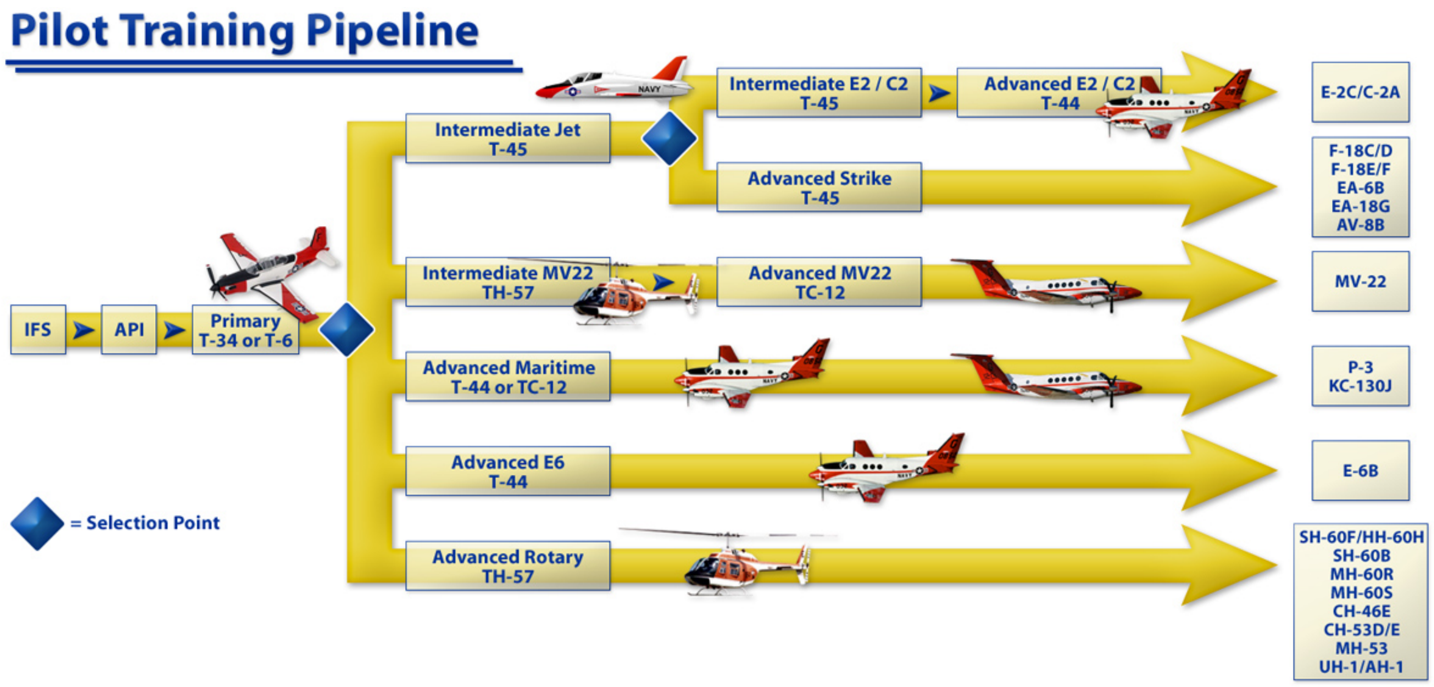


Figure 4. Naval Aviator Training Pipeline (From Chief of Naval Air Training 2012)

According to CNATRA's Assistant Chief of Staff for Resources (N8), the FY11 cost to train a Naval Aviator through the TRACOM Strike syllabus was just over $1.1 million dollars (personal communication, October 29, 2012). FRS costs would raise that number even higher before a Fleet aviator was produced.

3. Total Ownership Cost Optimization

a. Organizations and Objectives

The Director of Air Warfare (N88) initiated a Total Ownership Cost (TOC) optimization objective in 2011 to determine if efficiencies could be found in aviation training. Supporting this initiative, the Program Office for Undergraduate Flight Systems (PMA-273) worked with CNAF and CNATRA to redefine the training pipelines and associated syllabi. Their goal was simple—to provide the Fleet with a better trained aviator at a lower cost.

PMA-273 selected the Strike pipeline as the first training pipeline for analysis with the intent of applying the same methodology to the rest of the training pipelines after proven success. PMA-273 coordinated with NAWCTSD HP/ISD to set objectives for the TOC initiative. According to NAWCTSD HP/ISDs lead scientist, Dr. Joseph Sheehan, in a technical approach overview brief given in Naval Air Station Fallon, NV on June 12, 2012, the objectives were to:

- Develop a set of Fleet-supportive, generalizable skills
- Develop a process for understanding and evaluating the baseline training continuum for NAE skills within multiple training pipelines
- Develop a semi-automated system to predict the outcomes of proposed solutions
- Execute process on representative Type/Model/Series (T/M/S) aviation platforms.

The first two objectives have since been accomplished. This thesis focused on the third objective—to develop a semi-automated model capable of supporting analysis efforts. The model would explore the effects of re-allocating hours across the continuum on both cost and aircrew proficiency. According to Dr. Sheehan, the model would provide two main capabilities; determining the overall effect alterations in training hours may have on the overall proficiency of graduating aircrew; and identifying training combinations that satisfy the utilization of different platforms to achieve the desired cost and proficiency levels (Teleconference July 23, 2012).

b. New Grading Convention Supporting the Initiative

In a parallel effort, CNATRA was finalizing a new grading convention that provided the framework to understand and evaluate the baseline training system. CNATRA retired the old Navy Standard Score Grading (NSSG) convention, in favor of the new Multi-Service Pilot and NFO Training System (MPTS/MNTS).

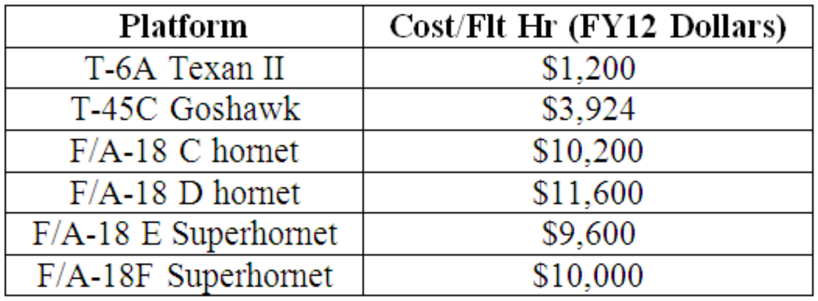
With the NSSG convention, grades were assigned to specific flight objectives but provided limited description of the overall proficiency gain of desired skills across the continuum. As such, when a student was flagged with a signal of difficulty, it was challenging to identify occurrences of related deficiencies in a timely manner (Chief of Naval Air Training 2012). In other words, skill proficiency gain was not traceable throughout the pipeline.

The MPTS/MNTS convention breaks each stage of training down into carefully designed training blocks which, in turn, incrementally build and refine a baseline set of required skills (Chief of Naval Air Training 2012). The training blocks satisfy learning objectives that support task lists and functions within each pipeline—ultimately building proficiency in the identified pipeline skills. The knowledge, skills, and abilities (KSAs) developed throughout the continuum are also monitored, allowing common threads in student deficiencies to be identified and tracked more effectively. The TOC optimization initiative leveraged the new modular framework of the training continuum and treats the training blocks as movable pieces—capable of being "downloaded" or "offloaded" across the training continuum.

c. Cost-reduction Techniques

If a training efficiency is to be considered, it must provide a reduction in the overall cost-to-train while not allowing skill proficiency to drop below desired levels.

According to U.S. Fleet Forces Command N02 representative CDR William Mallory, Table 1 shows the FY12 operating cost per flight hour in U.S. dollars of all platforms currently in the Strike pipeline (personal communication, September 4, 2012).



Platform	Cost/Flt Hr (FY12 Dollars)
T-6A Texan II	$1,200
T-45C Goshawk	$3,924
F/A-18 C hornet	$10,200
F/A-18 D hornet	$11,600
F/A-18 E Superhornet	$9,600
F/A-18F Superhornet	$10,000

Table 1. Cost Per Flight Hour of Platforms in the Strike Pipeline

Downloading is a simple concept – fly sorties in platforms that are cheaper to operate. In the Strike example, flights normally flown in an F/A-18 Hornet or Superhornet would be moved to a T-45C Goshawk or a T-6A Texan II to reduce the cost to complete them. These cost efficiencies don't come without a trade-off. Platforms with lower operational cost are typically platforms with less capability. To address this capability degradation, a downloaded sortie may require more flight hours in a lower cost platform to achieve the same KSA achievement.

Similar to downloading, the concept of offloading attempts to minimize cost while maintaining a required proficiency level, but accomplishes it by moving from the actual platform to a simulated platform. Once again, offloading efficiencies do not come without a price tag. Although significant advances have been made in modeling and simulation, there are some areas of tactical aviation that simulators still have trouble accurately replicating. Differences between simulator and aircraft may result in different rates of proficiency gain, requiring more repetition or longer hours in the simulator to compensate for lost flying time.

The TOC optimization initiative considers both of these techniques as critical components for optimizing training.

d. General Aviator Learning Equation (GALE) Methodology

NAWCTSD HP/ISD conducted a TOC workshop in April 2012 with the different TRACOMs and FRSs to gather data to support their methodology concept. Instructors and Subject Matter Experts (SMEs) were asked to provide high 90, low 10 and most likely values of proficiency provided by each block of flights. A high 90 value indicates the best proficiency gain value seen at most once out of ten times while the low 10 value indicates the worst proficiency gain value seen at most once out of ten times. A most likely value indicates the proficiency gain that the SME would expect to see in the average SNA based on the specific block.

Baseline phase entry and exit proficiency values based on the current syllabus were also obtained from pipeline SMEs during this event. Using the Strike syllabus as an example, the baseline values of zero proficiency were assigned to Primary entry proficiency values because SNAs would have no previous flight experience. However, upon completion, SNAs would be capable of achieving a certain proficiency level unique to each skill. NAWCTSD HP/ISD analyzed the SME input and provided a final expected exit proficiency value for each skill.

SMEs were then asked to quantify the proficiency loss that a typical SNA would suffer if he/she were to fly a downloaded sortie in a different platform. For example, if a specific air-to-air flight was flown in a T-45C instead of an F/A-18C, what percentage of proficiency would result? If not flown in the aircraft of choice, would some degradation in training occur? Twelve SMEs were polled and the resultant values were provided as a number known as a “media degrader value.” This value represents the percentage of proficiency one can expect to get if the flight in question is downloaded from one platform to another. Every building block across the pipeline was assigned media degrader values for every download scenario. These values are utilized by GALE methodology to reduce downloaded flights by an appropriate amount to provide the final reduced proficiency curve.

Additionally, there were “loss factors” provided by SMEs; these loss factors represented the amount of proficiency lost between phases in training. As an example, when a SNA leaves Primary flight training and enters Intermediate flight training, there is a loss of proficiency that occurs in the transition between a turbo prop aircraft and a jet aircraft. This degradation would set back the SNAs proficiency slightly, until more proficiency was attained by conducting training events. To keep the desktop model simple, the degradation was assumed to be constant between phases.

The resultant data sets were utilized by NAWCTSD HP/ISD to create the GALE proficiency curves. Simple mathematical manipulation provided the ability for GALE proficiency curves to be generated for every skill, across the pipeline. The general shape of the curve showing where proficiency is gained across the pipeline phases was based on skill proficiency entry and exit data. The shape of the curve within each phase is a result of the individual block contribution to the total phase proficiency gain.

Mathematically, the amount of proficiency gain ($P_{R_{\text{Block}}}$) due to each block is determined as the product of effectiveness and hours as a proportion of the total gain in the phase:

$$P_{R_{\text{Block}}} = \frac{(\text{Effectiveness Rating})(\text{Block Hours})}{P_{R_{\text{Phase Total}}}}$$

where $P_{R_{\text{Block}}}$ is the individual block proficiency contribution and $P_{R_{\text{Phase Total}}}$ is the total skill proficiency gained in the phase.

Additionally, overall phase proficiency, at any point, can be described as the cumulative sum of blocks' proficiencies prior to the evaluation point:

$$\text{Phase Proficiency} = P_{R_{\text{Block}}} + C_P$$

where C_P is the cumulative skill proficiency on the previous block. Combined, the equations form the basis of the baseline GALE proficiency curve.

Examples of two manually-created GALE proficiency curves were provided by NAWCTSD HP/ISD in a methodology briefing chart and are shown in Figure 5.

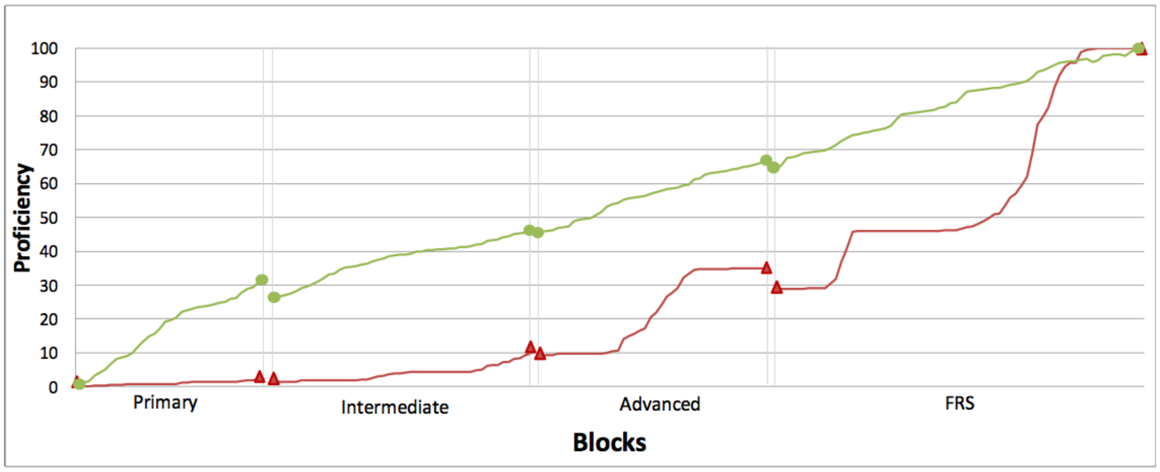


Figure 5. Example of GALE Proficiency Curves
(From Joseph Sheehan, pers. comm.)

Overall proficiency in each skill is plotted as a function of individual block proficiency contribution. The difference in general shape of the curves is observable—one skill, represented by the top line in Figure 5 (green line), displays a more linear proficiency gain across the pipeline, while the other, represented by the bottom line in Figure 5 (red line), displays a proficiency gain weighted heavily in the latter phases of the pipeline. This example depicts a couple main points of interest; all skills have a unique learning curve shape, based on when and where proficiency is gained; and although all pipeline skills are affected by altering an individual block, the effect on each skill could vary significantly.

What-if methodology is similar to the baseline methodology, except that it relies on hypothetical syllabus alterations to adjust the learning curve. The difference between the two curves represents the effect on skill proficiencies that specific syllabus manipulation causes. The variance in proficiency can be balanced with the variance in cost to then help determine if the hypothetical syllabus alteration provides the best value for the NAE.

C. STRIKE PIPELINE BREAKDOWN

It is imperative to understand the different phases of flight training that aircrew—Naval Aviators in the Strike pipeline in this case—complete in order to gain the appropriate KSAs necessary to become Fleet ready.

SNAs initially partake in Aviation Preflight Indoctrination, or API, in Pensacola FL. In API they are exposed to physical and academic challenges in areas including: engineering, aerodynamics, air navigation, air physiology, and water survival (Chief of Naval Air Training 2012).

Upon completion of API they enter their first flying phase, known as Primary flight training. Primary SNA training is conducted at three military installations:

- NAS Whiting Field, in Milton FL
- NAS Corpus Christi, in Corpus Christi TX
- Vance Air Force Base (AFB), in Enid OK.

Training lasts for approximately 22 weeks and includes the following in either the T-34 Turbomentor or the T-6A Texan II aircraft:

- Ground-based academics
- Simulators
- Flight Training

In Primary, the focus is on basic airmanship and aircraft control. Training consists of six stages: Familiarization (FAM), Basic Instruments (BI), Precision Aerobatics (PA), Formation (FORM), night FAM, and Radio Instruments (RI). Upon completion of primary, pipeline selection occurs. The selection point and available pipelines for SNAs are shown in Figure 5 on page 4.

In the Strike pipeline, SNAs proceed to the second phase of flight training commonly known as Intermediate. In the Intermediate phase, SNAs are trained in their first jet aircraft, the T-45C Goshawk, at one of two bases: NAS Kingsville in Texas, or NAS Meridian in Mississippi. SNAs learn the fundamentals of flying a faster, more agile aircraft and improve their proficiency levels in skills required by the Fleet.

Once competent in the new aircraft, they learn the fundamentals of turning an aircraft into a weapon system in the next phase known as Advanced. SNAs learn strike tactics, weapons delivery, air combat maneuvering, and receive their carrier qualification (Chief of Naval Air Training 2012). Upon completion of the Advanced syllabus, SNAs become designated Naval Aviators. At this point, Fleet platform selection occurs and the newly designated Naval Aviators proceed to their respective Fleet Replacement Squadrons.

At the FRS, Naval Aviators fly their assigned Fleet aircraft for the first time where they continue to add to the skill proficiencies built to date. In the FRS, they learn the intricacies of their Fleet aircraft, and receive an elementary education in operational ordinance delivery. Additionally, they are exposed to operational tactics and procedures in an effort to provide them with enough skill proficiency to be a contributing member in the operational Fleet.

A general understanding of the ongoing optimization efforts within Naval Aviation combined with a more detailed understanding of the training pipeline's intricacies helped when trying to conceptualize the model that was required to support the analysis effort. The next chapter focuses on the model concept stage to lay the appropriate groundwork for design and development efforts.

II. CONCEPT STAGE

A. ESTABLISHING CAPABILITY AND FUNCTIONALITY

1. System Capabilities

The process began with understanding what capabilities and functions must be provided by the model being designed. In this case, the model had to provide two user-defined capabilities provided by NAWCTSD HP/ISP:

- Determine the overall effect hour reductions or alterations may have on the overall proficiency of graduating aircrew
- Identify training combinations that satisfy the utilization of different platforms to achieve the desired cost and proficiency levels.

Next, the model's functions had to be identified and documented to ensure the capabilities could be delivered. To gain a deeper understanding of what was being modeled, a functional analysis on the Strike training pipeline was conducted prior to attempting the analysis on the model. This effort is documented in Appendix A.

2. NAPA Model Functional Analysis

Focusing on the model, the primary functional requirements needed to provide the desired capabilities are two-fold: the system must first be able to create a pipeline specific model representative of the desired training pipeline, and it must provide the ability to analyze that model given different user-defined scenarios. These were placed in context using an IDEF0 diagram (Figure 6).

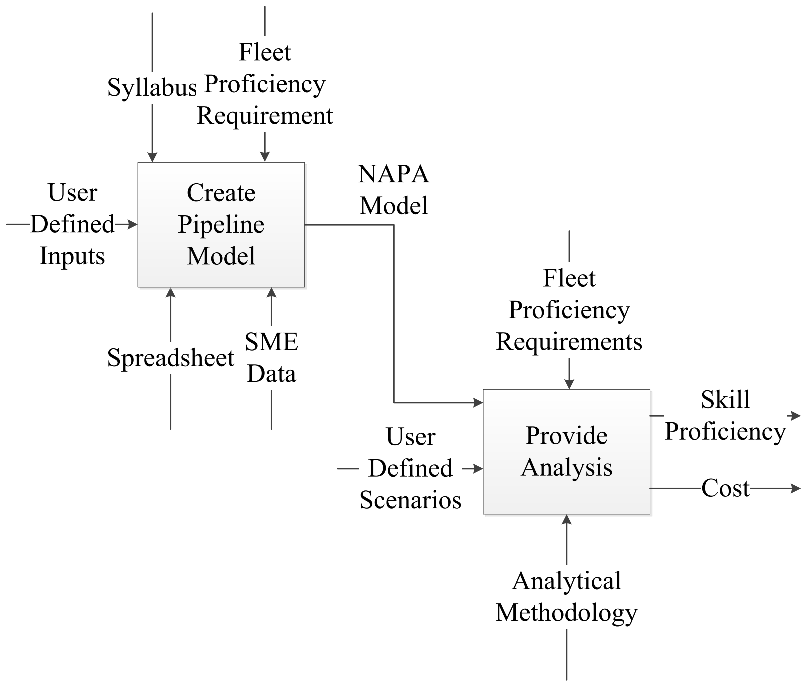


Figure 6. NAPA Model IDEF0

The model must be created by first accepting user-defined input data. The function of creating a pipeline specific model must be controlled by the training syllabus and the current Fleet proficiency requirement (both user identified). The function would utilize a spreadsheet or data manipulation program as well as available SME data as mechanisms to provide a computer-based model of a specific Naval Training pipeline.

In order to provide analysis, specific user-defined scenarios must be combined with the computer-based model, controlled once again, by Fleet proficiency requirements. The analytical methodology selected must provide the mechanism to facilitate scenario analysis, ultimately producing resultant cost and skill proficiencies.

Completing this portion of the analysis developed a deeper understanding of functional relationships, data requirements, and basic manufacturing framework of the model. However, not all functions of the model were represented yet. In order to produce a complete functional decomposition of the required system, consideration had to be given to how the model would interact with its intended environment. The additional analysis resulted in Figure 7, the final functional decomposition of the NAPA model.

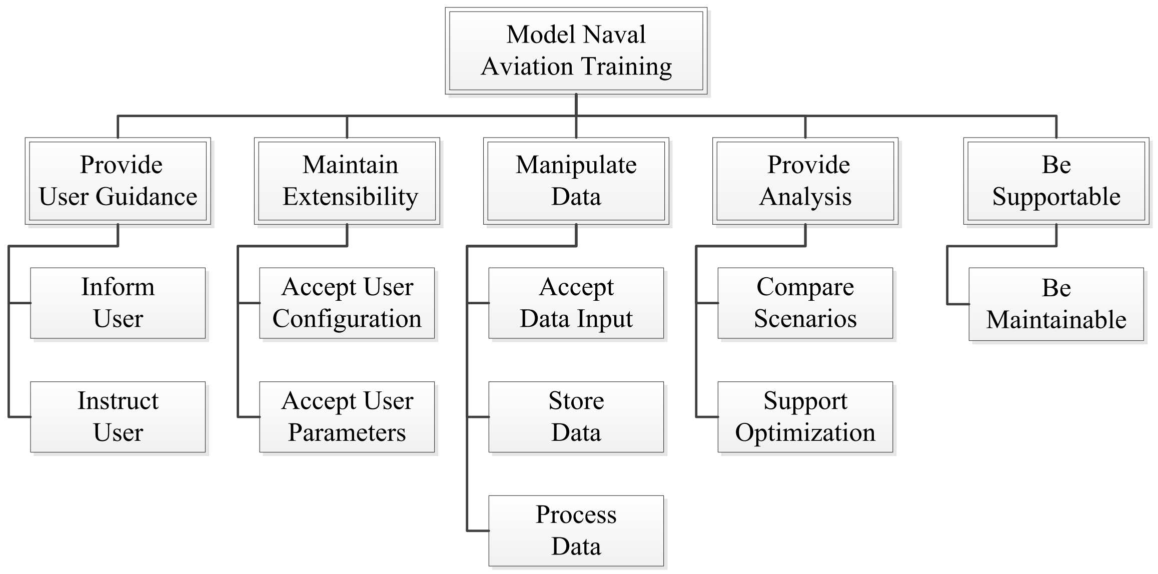


Figure 7. NAPA Model Functional Decomposition

There were five primary functions identified. Specifically, the model must:

- *Provide user guidance*. Inherent user guidance is critical to increase the overall usability of the model. Specifically, guidance can be informative or instructional.
- *Maintain Extensibility*. The model has to account for differences within each training pipeline. It must accept user configurations and parameters to account for those differences.
- *Manipulate Data*. The model must be able to accept, process, and store user-defined data sets in a format common to all pipelines.
- *Provide Analysis*. Obviously, the model must be able to provide the desired analysis. Specifically, it must be able to compare scenarios or specific alterations in training, as well as support optimization analysis of the training pipelines.
- *Be Supportable*. A supportable model can provide a longer service life. The maintainability of the model is also crucial to long-term supportability of the model.

B. REQUIREMENTS DEFINITION AND ANALYSIS

As the primary stakeholder and project sponsor, NAWCTSD HP/ISD provided a list of needs that the model should satisfy. The original needs statement is provided in Appendix B.

Since the TOC initiative was already in progress, transforming user needs into requirements did not require much iteration. A series of telephone calls and email exchanges with NAWCTSD HP/ISD personnel confirmed the top-level requirements the model must fulfill, as well as the system behaviors the model must display. Specifically, the model shall:

- Accept user-defined data sets
- Accurately represent user-selected pipeline
- Support variations of Basic Query: Provided defined linkages, how single or multiple training efficiencies alter expected levels of proficiency
- Be searchable and sortable
- Inform user of breached thresholds
- Allow user-defined parameters to dictate model baseline structure
- Incorporate live, real-time updating for scenario analysis
- Provide proficiency and cost outputs for analysis

Additionally, some top-level derived requirements were identified to provide the functionality identified in the functional decomposition. In particular, the model was required to:

- Be executable on any Navy Marine Corps Intranet (NMCI) computer
- Be operable by an individual with a baseline of knowledge in Naval Aviation training
- Be usable by an individual with a basic familiarity (input, output, copy, paste, and basic navigation / manipulation) with spreadsheet programs (e.g. Microsoft Excel or iWork Numbers)
- Be extensible (usable by all training pipelines)
- Be maintainable by an individual with above average computing skills (some programming experience in the programming language selected)

1. Software Constraints

Since any NMCI computer may be required to operate the model, programming software would be limited to the NMCI software library. This, combined with the requirement for the model to be spreadsheet-based, resulted in a constraint to use Microsoft Excel and its resident Visual Basic for Applications (VBA) programming capability for model creation. Additionally, since many aftermarket optimization applications require annual financial support, any optimization routines would have to be accomplished with the built-in optimization modeling system or Solver unless funding allowed otherwise.

2. Methodology Limitations

The model had to utilize NAWCTSD HP/ISP's GALE methodology to analyze training pipelines. This would allow the model to seamlessly integrate with the ongoing effort, providing immediate benefit to the primary stakeholder.

3. Data limitations

Although a significant amount of data was provided in support of this effort, offloading data was not available. NAWCTSD HP/ISD is still in the process of gathering data on moving blocks from flight platforms to simulator platforms. Until a complete set of data is available, downloading is the only initiative capable of being analyzed. As such, the model would be designed to support downloading analysis immediately, while maintaining the ability to incorporate offloading analysis in the future.

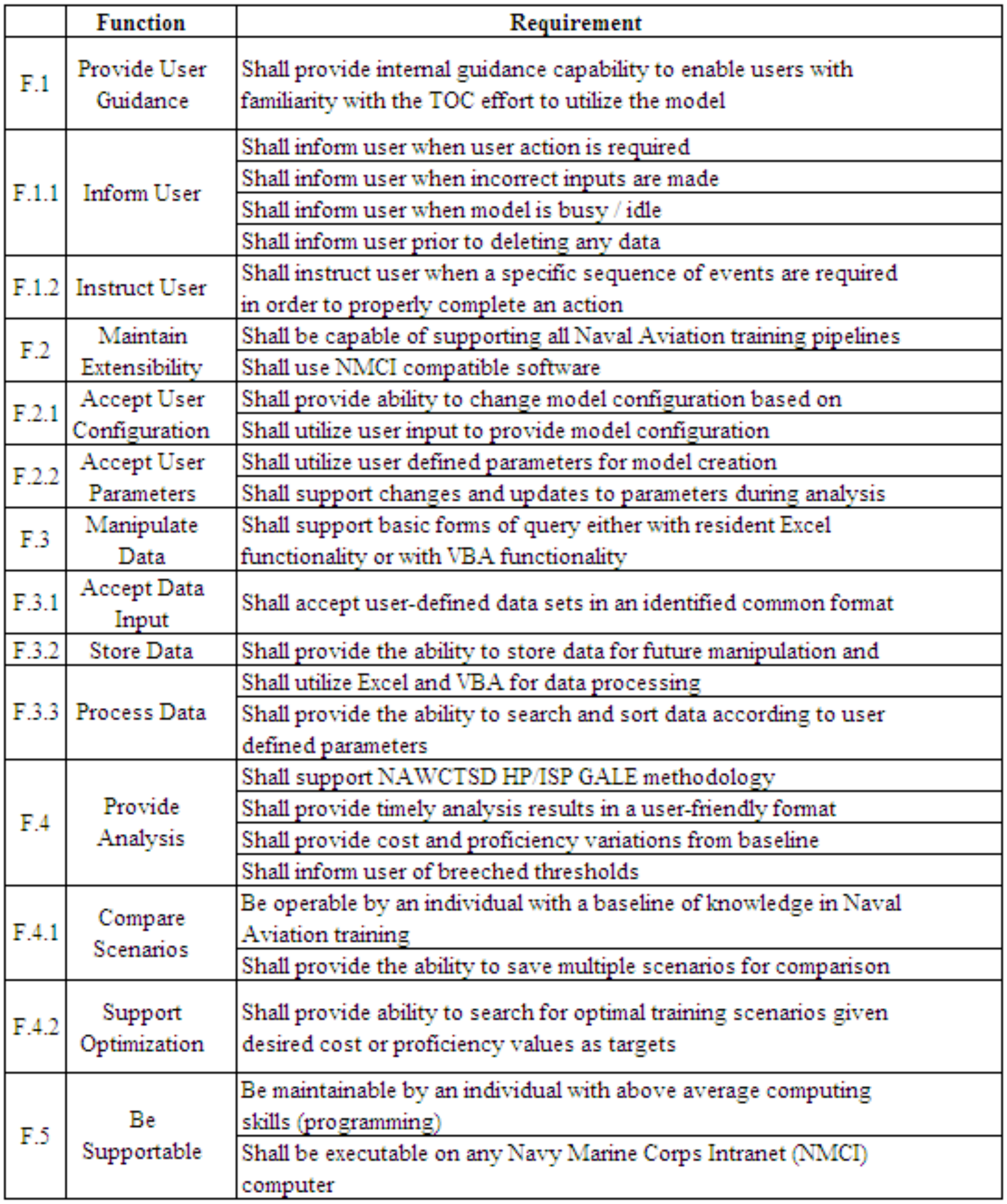
4. Assumptions

Along with constraints and limitations, assumptions made throughout the effort had to be captured for future reference. Table 2 lists the assumptions made to complete the modeling effort.

Assumption	Explanation
Input data will remain the same for all pipelines	In order to use the model with all pipelines, the input data must be similar in format and must have been gathered in the same context.
All pipelines utilize four main stages of training	Primary, Intermediate, Advanced, and the FRS are the phases common across all pipelines.
All training pipelines can be decomposed into skills and functions	Skills and functions must be identified for every pipeline in order to operate with the model.
Data values will change	Different pipelines will supply unique values for user-defined data. As such, the model must not be hardcoded.
GALE Methodology will be valid for all platforms	The NAWCTSD methodology can be applied to all training pipelines.
Model creation will not be done regularly, model manipulation will be.	Once the pipeline model is created, it should remain valid until a syllabus change or update is made. Evaluting scenarios of downloading or offloading within the syllabus will occur on a regular basis.

Table 2. Assumptions For Model Creation

Tying all requirements, limitations, assumptions, and constraints together helped develop a final list of requirements. A list of these requirements mapped to the identified functions is provided in Table 3.



	Function	Requirement
F.1	Provide User Guidance	Shall provide internal guidance capability to enable users with familiarity with the TOC effort to utilize the model
F.1.1	Inform User	Shall inform user when user action is required
		Shall inform user when incorrect inputs are made
		Shall inform user when model is busy / idle
		Shall inform user prior to deleting any data
F.1.2	Instruct User	Shall instruct user when a specific sequence of events are required in order to properly complete an action
F.2	Maintain Extensibility	Shall be capable of supporting all Naval Aviation training pipelines
		Shall use NMCI compatible software
F.2.1	Accept User Configuration	Shall provide ability to change model configuration based on
		Shall utilize user input to provide model configuration
F.2.2	Accept User Parameters	Shall utilize user defined parameters for model creation
		Shall support changes and updates to parameters during analysis
F.3	Manipulate Data	Shall support basic forms of query either with resident Excel functionality or with VBA functionality
F.3.1	Accept Data Input	Shall accept user-defined data sets in an identified common format
F.3.2	Store Data	Shall provide the ability to store data for future manipulation and
F.3.3	Process Data	Shall utilize Excel and VBA for data processing
		Shall provide the ability to search and sort data according to user defined parameters
F.4	Provide Analysis	Shall support NAWCTSD HP/ISP GALE methodology
		Shall provide timely analysis results in a user-friendly format
		Shall provide cost and proficiency variations from baseline
		Shall inform user of breeched thresholds
F.4.1	Compare Scenarios	Be operable by an individual with a baseline of knowledge in Naval Aviation training
		Shall provide the ability to save multiple scenarios for comparison
F.4.2	Support Optimization	Shall provide ability to search for optimal training scenarios given desired cost or proficiency values as targets
F.5	Be Supportable	Be maintainable by an individual with above average computing skills (programming)
		Shall be executable on any Navy Marine Corps Intranet (NMCI) computer

Table 3. Function / Requirement Mapping

After identifying and understanding the system’s requirements, a process model was selected to guide the remainder of the effort. The next section describes the model selection and modification process.

C. SELECTING A GUIDING PROCESS MODEL

The Systems Engineering (SE) process should be a frame of reference, tailored to the specific program in need (Blanchard and Fabrycky 2011). Figure 8 shows the tailored SE process implemented for the NAPA model design and prototype development.

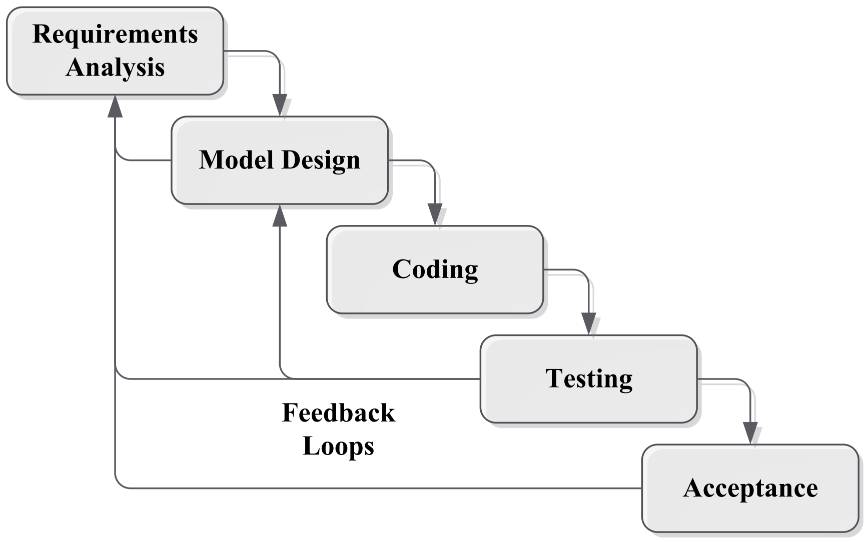


Figure 8. Tailored Waterfall Process For NAPA Modeling

A tailored waterfall process model was chosen as the reference mindset for overall model design—every attempt would be made to complete each phase prior to starting the next one. Ideally, each phase in the process is carried out to completion in sequence until the product is delivered (Blanchard and Fabrycky 2011). This type of model was selected because the user needs were stable, allowing the requirements analysis to be completed up front. Additionally, a stable schedule for completion allowed concrete planning of each phase. Iterations were anticipated early in the design phase. Model design would be complete prior to the coding effort. Developmental testing had to be accomplished while prototyping the model in the design phase, but once prototyping and coding were complete, it would be provided to NAWCTSD HP/ISD for final testing and acceptance.

With a guiding process in place, the final process in the conceptual stage was to establish a concept of operations.

D. OPERATIONAL CONCEPT

Two use cases were identified to help substantiate the operational concept; create a new pipeline training analysis model; and update an existing pipeline training analysis model.

1. Use-Case 1: Analyzing a Specific Training Scenario

In the first use-case, the user would be starting with an unpopulated model and would have to accomplish all steps of model creation to begin the process. After the model was created, a specific scenario would be input into the model and the resultant outputs (proficiency and cost) would be displayed. In order to create a new model, and conduct the analysis, the following actions must take place:

- The user defines the desired training pipeline by providing characteristics unique to it
- The user also provides the specific pipeline data to support GALE methodology and analysis in a common format
- The current syllabus is input to serve as a baseline from which to compare other training scenarios
- Training phase models are generated and GALE methodology is applied to provide the user with a baseline proficiency and cost output
- Once the model is created, the user inputs the training scenario for analysis
- The difference in proficiency is displayed to the user
- The cost difference is displayed to the user
- The user saves the scenario for future reference
- The user resets the model and repeats the scenario analysis until all scenarios are analyzed

2. Use-Case 2: Searching for a Training Scenario That Provides Desired Proficiency Levels

In this scenario, the user must search for the cost associated with providing the desired proficiency level. Additionally, the model has been previously created and must simply be reset before reuse:

- The user saves all unchanged data to reuse in the model

- The user resets the proficiency level thresholds to the desired levels
- The model reduces the likelihood of inadvertent data loss by informing user of the proposed changes and the consequences of the change
- After consent, the model makes the changes and updates the model and is ready for analysis
- The alternate training scenario is input into the model
- The model provides proficiency outputs and highlights troubled areas
- The user alters the training scenario until all desired proficiency levels are met
- The user compares baseline training cost, with alternative cost to train

Proposed data inputs and desired outputs for the operational concept, provided by NAWCTSD HP/ISD have been included in Appendix B. Understanding how the model would be utilized helped solidify all of the concept stage efforts and laid the groundwork necessary to transition into solution space, beginning with the design stage.

III. DESIGN AND DEVELOPMENT STAGE

A. MODEL DESIGN

1. Industry Best Practices for Application Design/Development

John Walkenbach, the author of *Excel 2010: Power Programming with VBA,* (2010) defines a spreadsheet application as "a file that is designed so that someone other than the developer can perform useful work without extensive training (101)." Although simple, this definition focused the design and development effort and provided a constant reminder not to over complicate the model.

In his book, Walkenbach also identifies characteristics of good spreadsheet applications. To paraphrase his list, a good application should:

- Help the end user perform a task that they may not be able to do otherwise
- Provide an appropriate solution to the problem
- Accomplish what it is supposed to
- Produce accurate results and be free of bugs
- Use appropriate and efficient methods and algorithms to accomplish its job
- Trap errors before the user is forced to deal with them
- Not allow the user to delete or modify important components accidentally (or intentionally)
- Provide a clear and consistent user interface so the user always knows how to proceed
- Properly document formulas, macros, and user interface elements to allow for subsequent changes if necessary
- Be designed so that it can be modified in simple ways without making major changes
- Provide an easily accessible "help" system that provides useful information on at least the major procedures
- Be portable and able to run on any system that has the proper software

Although not required, the NAPA model's design process utilized these characteristics as heuristics for application development.

2. Design Concept

The general concept during the design process was to ensure that the flow of information between the user and the model occurred in the correct order. A three-phase approach to creating the model was implemented. Phase one started with ensuring the correct pipeline configuration. Once the user was satisfied with the configuration, phase two leveraged user-defined inputs in phase one to create a pipeline specific parameters; parameters are defined as data that is configuration-based. After the model is framed, baseline pipeline training events are filled-in during phase three. These three phases combine to provide the canvas on which to apply the GALE methodology. Once GALE methodology is fused with pipeline specific data and training events, the actual analysis can be conducted.

3. Design Framework

Understanding the design concept facilitated the creation of a framework that identified the actual contents, required inputs, and desired outputs of the model. The framework helped establish data requirements for the model and create basic page layouts. Figure 9 graphically represents the flow of data as it enters from external nodes (yellow) and flows through the model design (gray nodes). Arrows represent information flow direction and identify what information is required ultimately to produce a representative pipeline model.

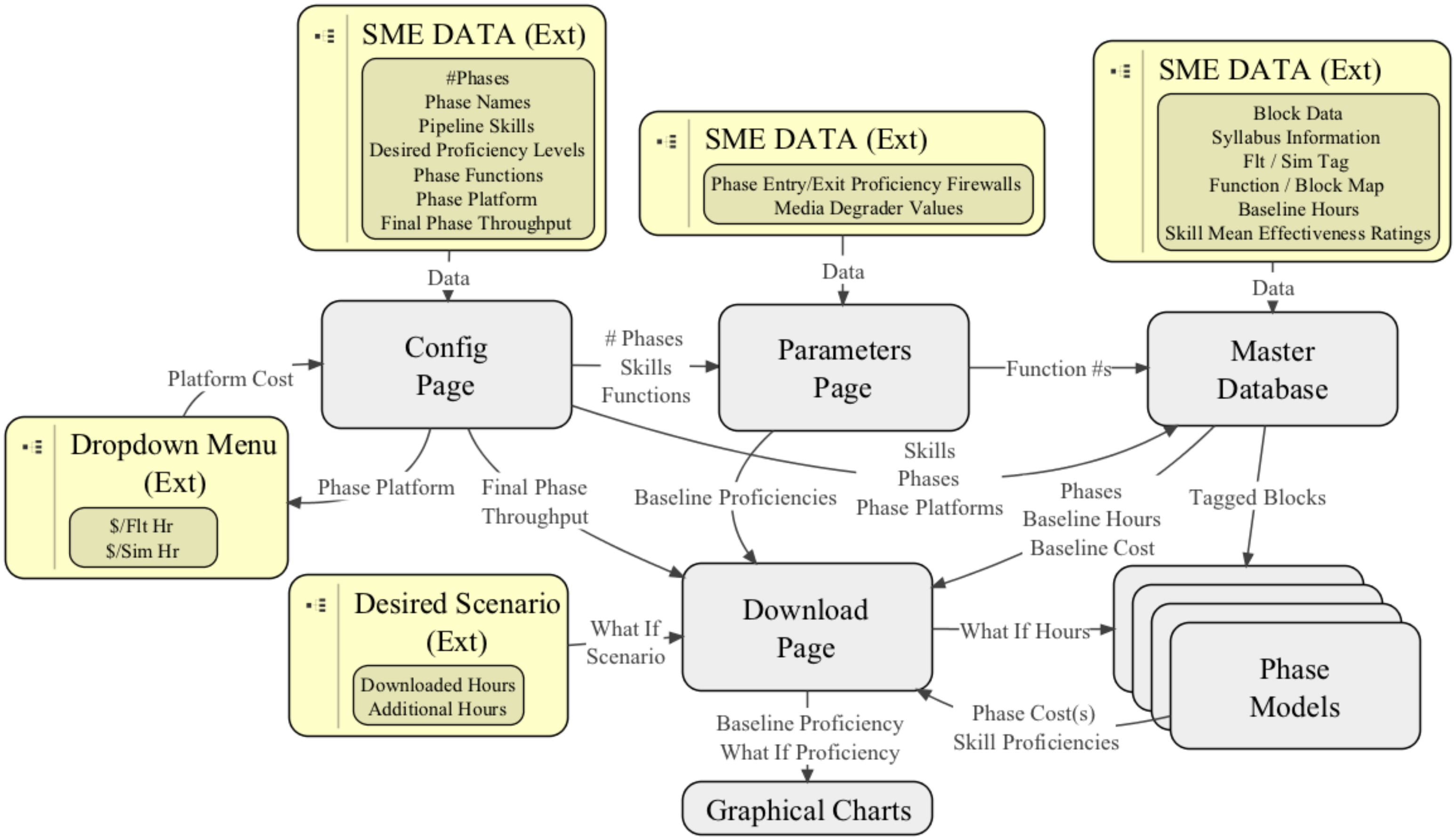


Figure 9. Basic Model Design Framework

The configuration ("Config") page must provide the initial user-model interaction during phase one. Users must enter the data required to construct a pipeline model. Basic information, including the number of phases; the phase names; pipeline skills and desired proficiency levels; phase platforms; and student throughput formulate the basis of the pipeline model.

In concept, a programmed macro transforms user-defined configurations into a "Parameters" page, where more data-gathering for the model would occur. The macro requires the input of the number of phases in training, what the phases were called, and the skills and functions associated with each phase to successfully create the page.

With the Parameters page created and filled-in, another page is created by macro. The master database or "MasterDB" page utilizes both Config and Parameter page inputs to create a basic table, capable of accepting pipeline specific data from the user. All phase models are then based on the MasterDB design.

Conceptually, the phase models serve as the location where most calculations occur. Each phase model, in turn, must provide baseline proficiency and cost values, as well as what-if proficiency and cost values to facilitate a comparison between the two. To support a single page input-output design, all phase models feed the "Download" page, where user-defined alterations in training can be immediately analyzed in terms of cost and skill proficiency.

4. Coding

Designing the code to be modular is commonly a good practice—it makes it scalable, extendable, and supportable. However, given the programming language, and the model framework established, an imperative programming style seemed appropriate to govern the code design. With imperative programming, individual subroutines alter the state of the model, requiring them to be run in a specific order for the model to be created properly. This supports the page / macro concept designed in the framework.

However, designing modularity into the code was not abandoned. If aspects of modularity could be integrated into the design in it would improve the maintainability, scalability, and extendibility of the model. To accomplish this:

- Functions that would be used multiple times need to be established as external functions accessible by all subroutines
- Variables and constants must be defined up-front, and referenced throughout the coding process to reduce efforts required to update or alter the model
- Individual subroutines must be defined by logical boundaries in model functionality
- The required inputs and outputs of each subroutine need to be captured to ensure future maintenance efforts can alter a subroutine without breaking the models overall functionality

Together, the design concept, model framework, and code design provided a roadmap for model development.

B. MODEL DEVELOPMENT

Two prototypes were constructed to verify model design concepts and application of methodology prior to final model development. The first prototype was a proof of concept—it focused on developing the proper spreadsheet functionality to support the required analysis. It was not designed to provide all of the required functions. The model was kept at a manageable size (10 blocks per phase instead of the typical 30–50 blocks). Controlled values were used in place of real data to facilitate rapid troubleshooting. Spreadsheet design and functionality were hard-coded to increase the speed of prototyping. Original outputs and screenshots of the first prototype are included in Appendix C. After the basic structure and logic of the design was verified, the focus turned to verification of model outputs.

The second prototype utilized the basic structure and logic of the design and applied it to actual data sets provided for the Strike syllabus. Outputs were generated and verified with NAWCTSD HP/ISD personnel. Two main insights were found as a result of this effort:

- The outputs of the model were consistent with NAWCTSD HP/ISD manual calculations.

- An after-market solver platform would be required to support training optimization efforts.

The second insight caused a re-evaluation of requirements, ultimately leading to stakeholders agreeing that pursuing the optimization capability during this effort was cost-prohibitive. As such, the requirement to provide training optimization was removed. The design concept, framework, and verified developmental efforts were combined to create the final NAPA model. The model was automated to guide a user through the input pages, create the training phase models based on those inputs, and support analysis efforts after creation. VBA was used to automate Excel functionality. Inexperience with the VBA programming environment resulted in fragile code. To help address this, William Evans, an Operations Research student at NPS with a Computer Science undergraduate degree, provided assistance to strengthen the code design. Code strengthening efforts included:

- Generalizing the code by removing "magic" number references (constant values that are hard-coded into the application without explanation)
- Making use of global variables to improve the readability of the code and ease the maintainability of the model
- Increasing the speed of the subroutines by utilizing more efficient coding procedures and practices
- Increasing model extensibility by allowing more user-defined parameters
- Providing in-application trouble shooting capability for the future maintainability of the code

Once code strengthening efforts were complete, the model was ready for final verification. A complete description of the NAPA model is provided in Appendix D. Complete VBA code can be found in Appendix E.

C. VERIFICATION AND VALIDATION

1. Verifying Requirements Were Met

Requirements and functions were mapped to final design elements of the NAPA model to ensure that no orphan requirements remained. The complete mapping of requirement to model element is provided in Appendix F.

To summarize, every requirement was satisfied, save one—optimization of training could not be provided due to software limitations. Since optimization of training was a user requirement, the architecture of the final model was designed to preserve the capability to integrate it in the future.

2. Verifying Model Performance

Previous coordination with the primary stakeholders during prototyping provided verification for the logic, design, and outputs of the model. The focus of this effort was verifying functionality added to address the overall extensibility of the model. Due to the size of the model, exhaustive checks were not feasible. Spot-checking different aspects of automation were conducted to ensure the model was performing as intended. These spot-checks, combined with properly verified outputs provided the confidence necessary to proceed with validation efforts.

3. Stakeholder Model Validation

NAWCTSD HP/ISD conducted the model validation efforts. They supported an active request by CNATRA to analyze a potential reduction in Primary phase flight hours in the Rotary training syllabus. Knowing that a reduction of flight hours would result in some proficiency degradation, CNATRA wanted to identify the difference in final proficiency if they chose different areas in the syllabus to take the hours from. Two scenarios were proposed:

- A reduction of 8.5 hours in PA flights in the Primary phase
- A reduction of 8.5 hours late in the RI flights in the Primary phase

NAWCTSD HP/ISD populated the NAPA model with the Rotary training syllabus data and utilized it to analyze both scenarios. The NAPA model's outputs allowed NAWCTSD HP/ISD to quickly identify that both scenarios significantly impacted the final proficiencies of four out of the eight skills in the Rotary training pipeline. This enabled NAWCTSD HP/ISD to inform CNATRA decision-makers about the potential impacts of their alternatives. NAWCTSD HP/ISD confirmed the outputs of the model in a final verification effort with their manual calculation methodology to ensure the outputs were accurate.

Once validated, the NAPA model was ready for delivery. NAWCTSD HP/ISD accepted ownership of the NAPA model in March of 2013, and currently utilizes it for desktop analysis efforts.

IV. CONCLUSIONS

The application of the SE process provided significant insight throughout the development of the NAPA model; it helped lay a solid foundation for the effort by concentrating on user needs and requirements; it provided the tools necessary to identify the model's architecture that would support those requirements; and it helped design and develop a working model that satisfied those requirements.

Success of the effort can ultimately be determined by the ability of the model to provide the capabilities desired by the user. Specifically, the model had to:

- Determine the overall effect training hour reductions or alterations may have on the overall proficiency of graduating aircrew
- Identify training combinations that satisfy utilization of different platforms to achieve the desired cost and proficiency levels.

The model easily identifies proficiency alterations as a result of training hour reductions within the syllabus. Additionally, it allows users to analyze different phase / platform / training hour combinations to achieve desired levels of cost and proficiency. Based on providing user capabilities, the effort was a success.

Another way to determine the "success" of the effort is to ask, "what might have been overlooked had a systems approach not been applied?" Three potential oversights were identified:

- The model may have been designed to support the Strike pipeline only, and not all other training pipelines
- The model may have been developed with a software suite not supportable logistically
- The model may have been developed with architecture that did not support expansion of analysis capabilities

The first potential oversight sounds almost trivial, but the effort taken to functionally decompose the system placed extensibility at the requirement forefront. Subsequently, the design for extensibility received the attention it needed throughout the process. Artifacts of this requirement are apparent in the architecture, design, and prototyping efforts taken to develop a properly extensible model. Early emphasis on this requirement resulted in a model that can be used by any training pipeline, without any modification. This capability was demonstrated during validation efforts where NAWCTSD HP/ISD analyzed a different pipeline (Rotary training) with the NAPA model.

The second potential oversight was also addressed in the requirements analysis process. Initially, after-market solving platforms were being discussed as potential suitable software suites. However, upon realizing that the model required NMCI compatibility and that user population may be diverse, the software constraint drove follow-on design and development efforts resulting in the use of an off-the-shelf product: Microsoft Excel. Once again, had this not been identified early in the process, significant re-work may have been necessary.

Finally, the third potential oversight was addressed by recognizing the desire for future analysis capabilities and incorporating them into the architecture and design of the model. Realizing that after-market solvers may be utilized someday, the framework of the model was designed to accommodate typical optimization software requirements. With little programming re-work, the model can be run on after-market Excel-based solvers such as FrontlineSolver's *Risk Solver Pro* or Real Options Valuation, Incorporated's *Risk Simulator*. Additionally, the lack of data to support the offloading effort made supporting that portion of the analysis impossible. However, re-programming the final model to support that effort can be accomplished by simply following the framework already in place.

Viewing this modeling effort through the eyes of a systems thinker increased the model's performance on many different levels. But not everything can be considered successful. Part of an honest analysis of an effort includes determining what could have been done better. In retrospect, providing a detailed requirements traceability verification matrix (RVTM) would have made the verification and validation of the model much easier. This effort mapped elements of the model to requirements to ensure no orphans existed, but it did not provide a threshold and objective against which to verify. In this case, verification was accomplished by comparison of manual GALE calculations with the NAPA model outputs and the primary stakeholder's feedback served to validate the model.

Ultimately, the SE process worked. It helped provide a desktop model, capable of facilitating the analysis of Naval Aviation training to NAWCTSD HP/ISD. Provided in the context of the Strike training pipeline, this model met the stakeholder need and will hopefully continue to provide analytical service for years to come.

THIS PAGE INTENTIONALLY LEFT BLANK

V. AREAS FOR FURTHER RESEARCH

After developing the current model, recommendations can be made in an effort to improve the overall quality of information provided.

A. SUPPORT OF OFFLOADING

Supporting the other efficiency recognized during this analysis is crucial to seeing the "entire" training picture. The inability to incorporate offloading into the model can be eliminated as soon as data becomes available to support the effort. Completion of this additional effort will significantly improve the quality of information provided to decision-makers.

B. TRAINING OPTIMIZATION

The model does not support the optimization of training, but rather it provides a tool to determine whether or not a specific scenario satisfies the cost and proficiency requirements set by the user. In short, there could be more efficient ways to achieve the same output, but significant effort has to be exerted to achieve it through this tool. Having a tool that can look for optimal solutions is something that can be investigated in the future. Should money become available to support after-market solvers, efforts can be focused on adapting the NAPA model to accomplish this task.

C. ALTERNATIVE METHODOLOGIES FOR ANALYSIS

Capability frontiers may be looked at to determine what is possible given different input parameters. Finding feasible solutions that satisfice, or provide an acceptable solution, rather than optimize the training pipeline may allow decision-makers to make cost-comparisons and select a unique path toward fulfilling the Fleet proficiency requirements.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: FUNCTIONAL ANALYSIS OF NAVAL AVIATION TRAINING (STRIKE PIPELINE)

From a system perspective, Naval Aviation training can be described in terms of inputs, outputs, controls, and mechanisms. At the highest level, the function of training Naval Aviators converts students (inputs) into Naval Aviators and NFOs (outputs), by instructors through the use of training aircraft, simulators, and facilities (mechanisms), via the training syllabi to meet proficiency requirements set by the Fleet (controls). These attributes were captured in the IDEF0 model of Naval Aviation training shown in Figure 10.

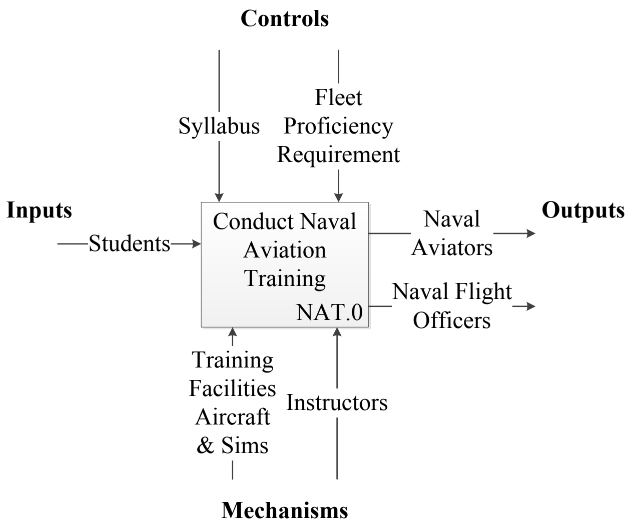


Figure 10. Top Level IDEF0 Model of Naval Aviation Training

The next step was to analyze the Strike training pipeline specifically. Figure 11 provides a more detailed IDEF0 model of a specific (Strike) training pipeline.

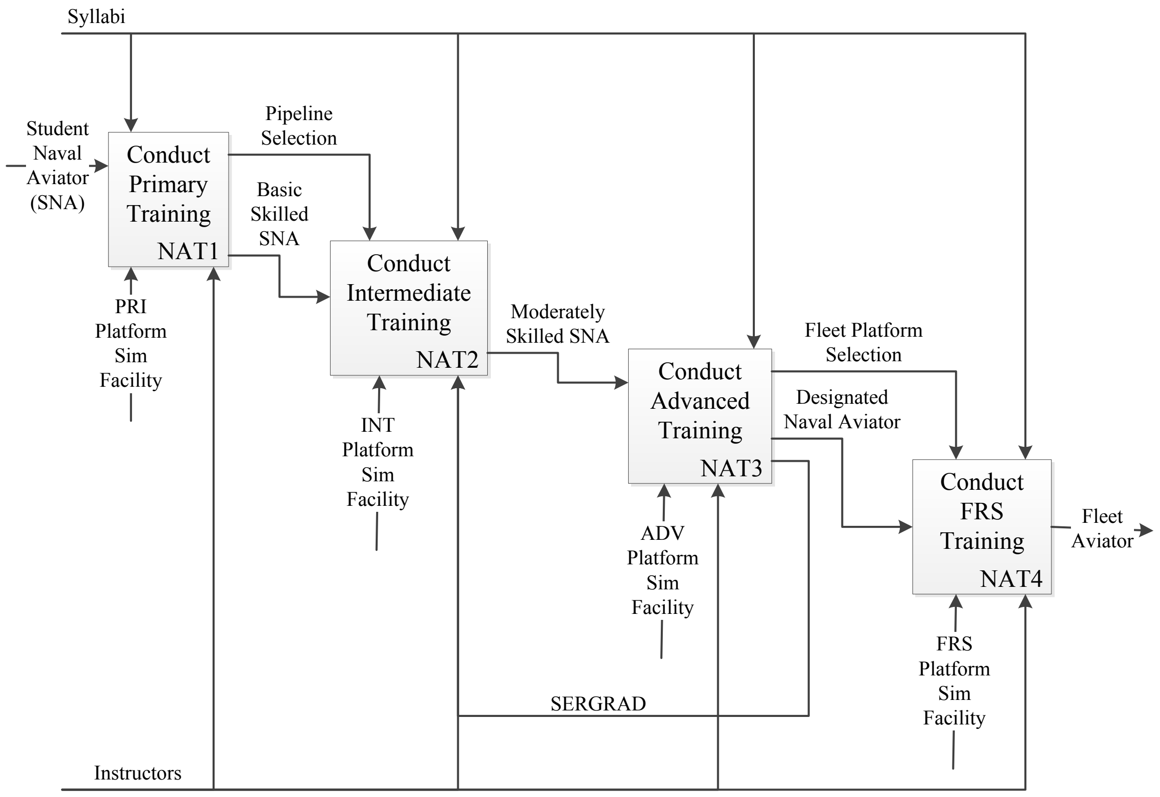


Figure 11. IDEF0 Detailed Model of the Strike Pipeline for Naval Aviators

The function of conducting Primary training transforms Naval officers selected as SNAs (inputs), into SNAs with basic flight skill proficiencies and knowledge of which training pipeline they will follow (outputs) controlled by the Primary syllabus, using instructors and the Primary platform, simulators, and Primary facilities as mechanisms.

Intermediate flight training converts SNAs with basic flight skill proficiencies into one with proficiency increases in skills desired by their selected pipeline. Controls during the Intermediate function include the specific pipeline selected and the intermediate syllabus associated with that pipeline. Mechanisms for intermediate include the instructors and the pipeline specific platform, associated simulators, and facilities that instruction will take place in. A mechanism unique to the Intermediate and Advanced training functions is the SERGRAD. A SERGRAD is a selectively retained graduate who displays enough skill proficiency through training to immediately be asked to join the instructor cadre before proceeding onto the FRS. A SERGRAD can be a mechanism for either Intermediate or Advanced functions.

The Advanced training function finally transforms a SNA (input) into a designated Naval Aviator with a specific Fleet platform to operate (outputs). Similar controls and mechanisms help facilitate this function.

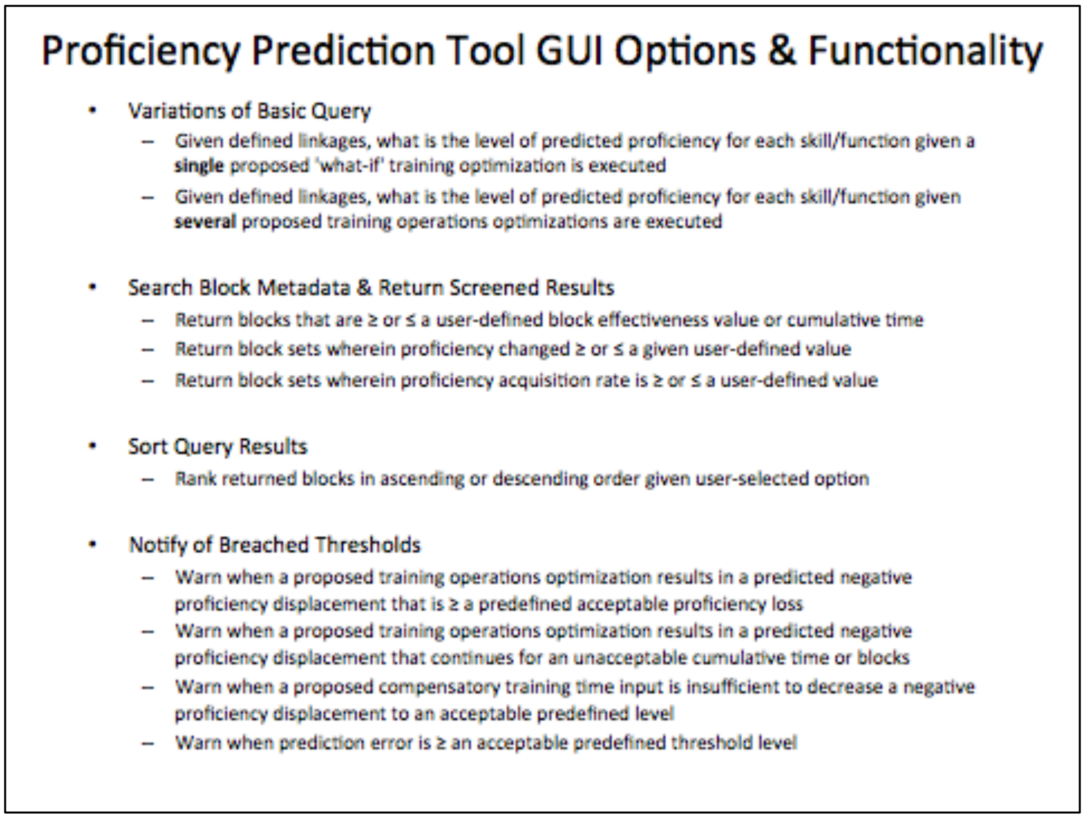
The FRS function transforms a newly designated Naval Aviator into a Fleet Aviator using mechanisms similar to previous functions. The FRS training function is controlled by the selected Fleet platform and FRS syllabus associated with it.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: ORIGINAL USER NEEDS STATEMENT AND INITIAL CONCEPT OF OPERATIONS

A. USER NEEDS

On July 23, 2012, in email communication, NAWCTSD HP/ISDs lead scientist, Dr. Joseph Sheehan provided a PowerPoint slide that captured initial stakeholder needs for the effort (Figure 12).



Proficiency Prediction Tool GUI Options & Functionality

- Variations of Basic Query
  - Given defined linkages, what is the level of predicted proficiency for each skill/function given a **single** proposed 'what-if' training optimization is executed
  - Given defined linkages, what is the level of predicted proficiency for each skill/function given **several** proposed training operations optimizations are executed
- Search Block Metadata & Return Screened Results
  - Return blocks that are ≥ or ≤ a user-defined block effectiveness value or cumulative time
  - Return block sets wherein proficiency changed ≥ or ≤ a given user-defined value
  - Return block sets wherein proficiency acquisition rate is ≥ or ≤ a user-defined value
- Sort Query Results
  - Rank returned blocks in ascending or descending order given user-selected option
- Notify of Breached Thresholds
  - Warn when a proposed training operations optimization results in a predicted negative proficiency displacement that is ≥ a predefined acceptable proficiency loss
  - Warn when a proposed training operations optimization results in a predicted negative proficiency displacement that continues for an unacceptable cumulative time or blocks
  - Warn when a proposed compensatory training time input is insufficient to decrease a negative proficiency displacement to an acceptable predefined level
  - Warn when prediction error is ≥ an acceptable predefined threshold level

Figure 12. Original Stakeholder Needs Documentation (From Joseph Sheehan, pers. comm.)

To expand on the slide:

- *Variations of Basic Query:* The users wanted the model to be capable of analyzing individual downloading / offloading efforts as well as the cumulative effect of multiple downloading / offloading efforts.
- *Search Block Metadata and Return Screened Results:* Having the data in a sortable list will allow the users to see which blocks have the highest rate

of return. Additionally, they can see where the most hours are currently allocated.

- *Sort Query Results:* Users want to be able to rank the results
- *Notify of Breached Thresholds:* Users want to be alerted if proficiency values drop below defined thresholds, or when syllabus alterations result in reduced proficiency.

B. OPERATIONAL CONCEPT: DATA FLOW

Use-case scenarios captured the high-level concepts of NAWCTSD HP/ISDs vision of how the NAPA model would be utilized. Additional detail of the operational concept, in terms of what inputs are required, as well as what specific outputs are desired are provided in this section. Figure 13 shows the desired inputs and outputs of the model.

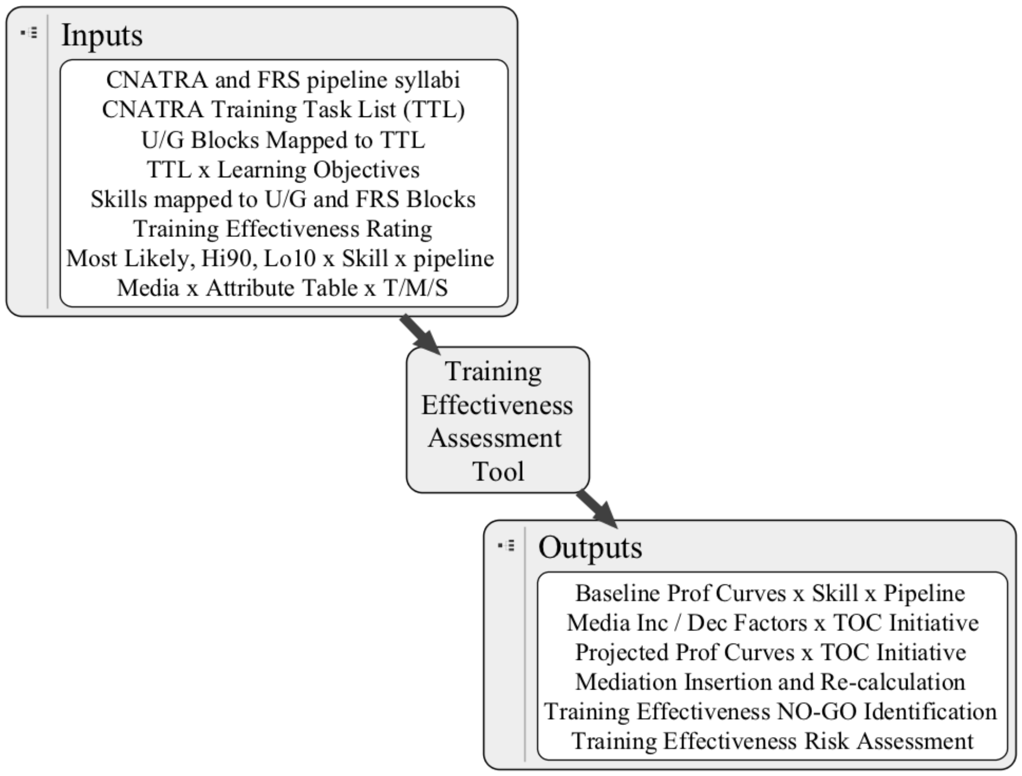

Inputs
CNATRA and FRS pipeline syllabi
CNATRA Training Task List (TTL)
U/G Blocks Mapped to TTL
TTL x Learning Objectives
Skills mapped to U/G and FRS Blocks
Training Effectiveness Rating
Most Likely, Hi90, Lo10 x Skill x pipeline
Media x Attribute Table x T/M/S
Training Effectiveness Assessment Tool
Outputs
Baseline Prof Curves x Skill x Pipeline
Media Inc / Dec Factors x TOC Initiative
Projected Prof Curves x TOC Initiative
Mediation Insertion and Re-calculation
Training Effectiveness NO-GO Identification
Training Effectiveness Risk Assessment

Figure 13. User Concept of Operations (After Joseph Sheehan, 2012 pers. comm.)

Inputs include the training syllabus, broken down in to training blocks that support a Training Task List (TTL). The blocks are mapped through learning objectives and skills and provided to the model as a vetted product. The blocks are also assigned a training effectiveness rating for each skill by SMEs. Media degrader values are also provided as a way to reduce the effectiveness rating of a block if its moved from the original platform to one of lower capability.

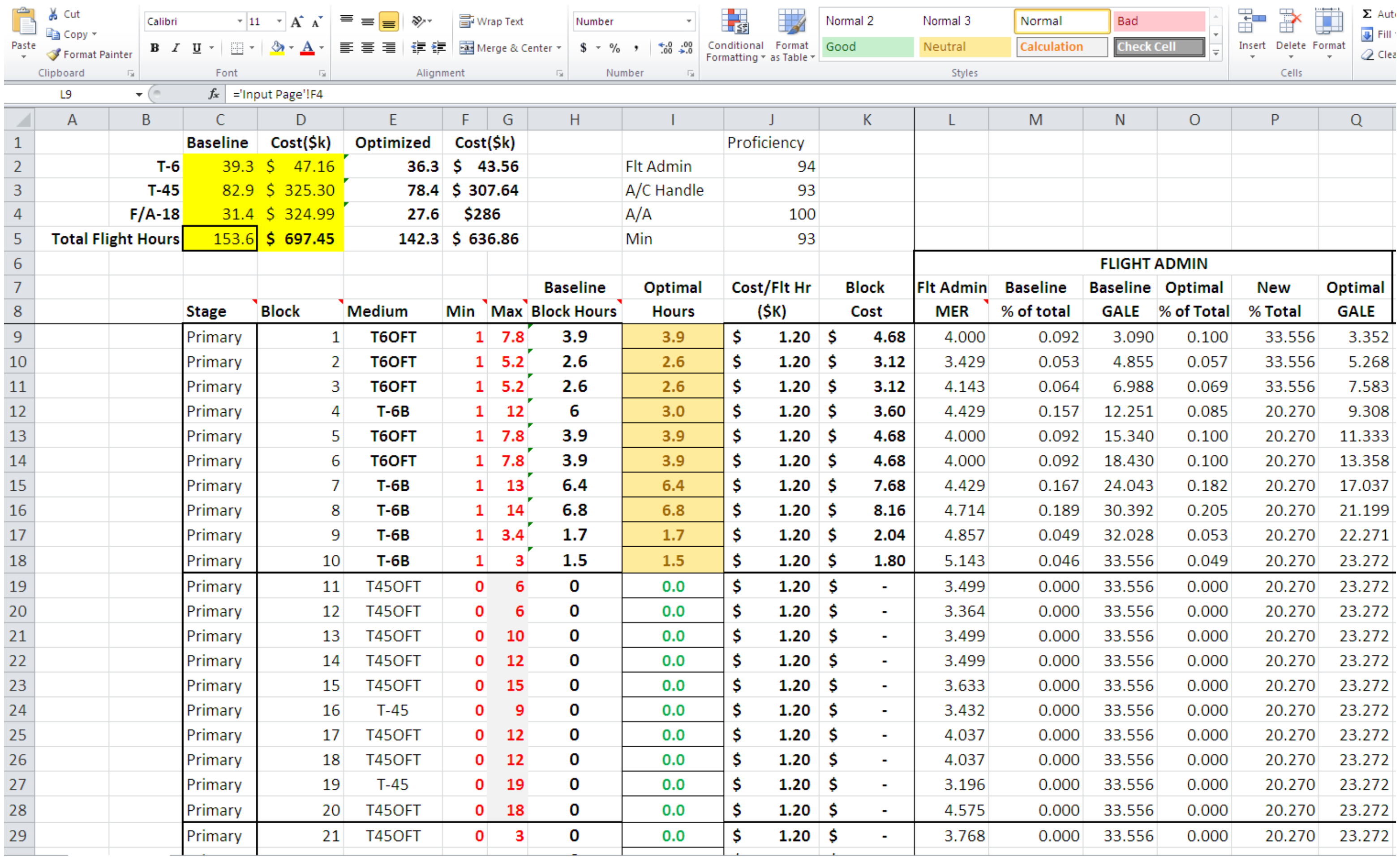
Outputs include a baseline proficiency curve for each skill, projected proficiency curves for the altered syllabus, and notification of any undesirable results based on user-defined parameters.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: MODEL PROTOTYPES

A. THE FIRST PROTOTYPE

The first prototype was built entirely in Excel, to provide proof-of-concept functionality and GALE methodology support. Specifically, the design included only three out of the eight skills for the Strike training syllabus; flight admin, aircraft handling, and air-to-air. The prototype utilized only ten blocks per phase, and included an attempt at optimization functionality. A partial screenshot example of the first prototype layout can be seen in Figure 14.



L9 ='Input Page'!F4

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1			Baseline	Cost($k)	Optimized	Cost($k)				Proficiency							
2		T-6	39.3	$ 47.16	36.3	$ 43.56			Flt Admin	94							
3		T-45	82.9	$ 325.30	78.4	$ 307.64			A/C Handle	93							
4		F/A-18	31.4	$ 324.99	27.6	$286			A/A	100							
5		Total Flight Hours	153.6	$ 697.45	142.3	$ 636.86			Min	93							
6												FLIGHT ADMIN					
7								Baseline	Optimal	Cost/Flt Hr	Block	Flt Admin	Baseline	Baseline	Optimal	New	Optimal
8			Stage	Block	Medium	Min	Max	Block Hours	Hours	($K)	Cost	MER	% of total	GALE	% of Total	% Total	GALE
9			Primary	1	T6OFT	1	7.8	3.9	3.9	$ 1.20	$ 4.68	4.000	0.092	3.090	0.100	33.556	3.352
10			Primary	2	T6OFT	1	5.2	2.6	2.6	$ 1.20	$ 3.12	3.429	0.053	4.855	0.057	33.556	5.268
11			Primary	3	T6OFT	1	5.2	2.6	2.6	$ 1.20	$ 3.12	4.143	0.064	6.988	0.069	33.556	7.583
12			Primary	4	T-6B	1	12	6	3.0	$ 1.20	$ 3.60	4.429	0.157	12.251	0.085	20.270	9.308
13			Primary	5	T6OFT	1	7.8	3.9	3.9	$ 1.20	$ 4.68	4.000	0.092	15.340	0.100	20.270	11.333
14			Primary	6	T6OFT	1	7.8	3.9	3.9	$ 1.20	$ 4.68	4.000	0.092	18.430	0.100	20.270	13.358
15			Primary	7	T-6B	1	13	6.4	6.4	$ 1.20	$ 7.68	4.429	0.167	24.043	0.182	20.270	17.037
16			Primary	8	T-6B	1	14	6.8	6.8	$ 1.20	$ 8.16	4.714	0.189	30.392	0.205	20.270	21.199
17			Primary	9	T-6B	1	3.4	1.7	1.7	$ 1.20	$ 2.04	4.857	0.049	32.028	0.053	20.270	22.271
18			Primary	10	T-6B	1	3	1.5	1.5	$ 1.20	$ 1.80	5.143	0.046	33.556	0.049	20.270	23.272
19			Primary	11	T45OFT	0	6	0	0.0	$ 1.20	$ -	3.499	0.000	33.556	0.000	20.270	23.272
20			Primary	12	T45OFT	0	6	0	0.0	$ 1.20	$ -	3.364	0.000	33.556	0.000	20.270	23.272
21			Primary	13	T45OFT	0	10	0	0.0	$ 1.20	$ -	3.499	0.000	33.556	0.000	20.270	23.272
22			Primary	14	T45OFT	0	12	0	0.0	$ 1.20	$ -	3.499	0.000	33.556	0.000	20.270	23.272
23			Primary	15	T45OFT	0	15	0	0.0	$ 1.20	$ -	3.633	0.000	33.556	0.000	20.270	23.272
24			Primary	16	T-45	0	9	0	0.0	$ 1.20	$ -	3.432	0.000	33.556	0.000	20.270	23.272
25			Primary	17	T45OFT	0	12	0	0.0	$ 1.20	$ -	4.037	0.000	33.556	0.000	20.270	23.272
26			Primary	18	T45OFT	0	12	0	0.0	$ 1.20	$ -	4.037	0.000	33.556	0.000	20.270	23.272
27			Primary	19	T-45	0	19	0	0.0	$ 1.20	$ -	3.196	0.000	33.556	0.000	20.270	23.272
28			Primary	20	T45OFT	0	18	0	0.0	$ 1.20	$ -	4.575	0.000	33.556	0.000	20.270	23.272
29			Primary	21	T45OFT	0	3	0	0.0	$ 1.20	$ -	3.768	0.000	33.556	0.000	20.270	23.272

Figure 14. Example of First Prototype Layout

The minimum, maximum, and optimal hours blocks were included in the first prototype to facilitate the optimization functionality of the model. Minimum hours would be based on a SME provided mandate to conduct a certain block of training—if the block were mandatory, a minimum of one hour would be required; however, if it were a downloaded flight, zero would be a possibility. Max hours were hard-coded to twice the baseline hours to cap the amount of proficiency one can achieve in a particular phase block—realistic scenarios would not allow a student to remain in a specific block for an unlimited amount of time.

The first prototype provided graphical output of the baseline and reduced hour proficiency for pilots. An example of the graphical output of the first prototype is shown in Figure 15.

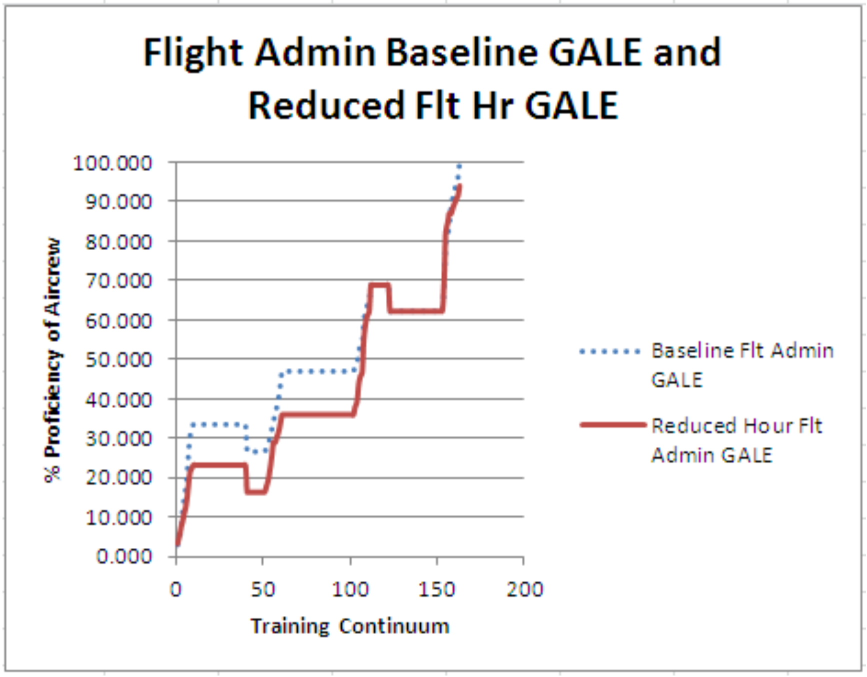
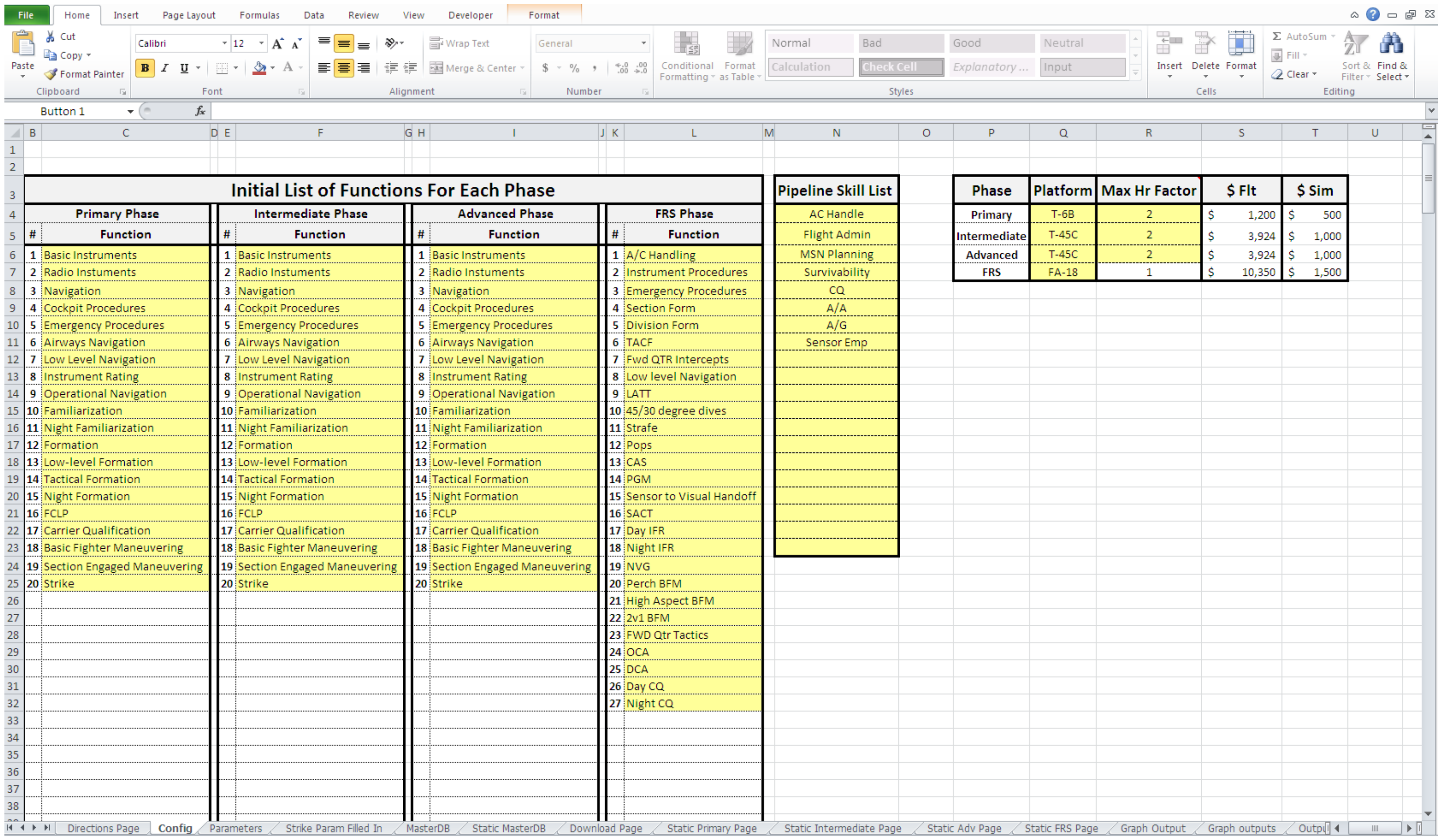


Figure 15. First Prototype Output Example

The goal of the first prototype was not to provide actual analysis, but rather to verify that the modeling methodology. The first prototype was provided to NAWCTSD HP/ISD for final analysis and approval. Upon acceptance of the model methodology, the second prototype effort began.

B. THE SECOND PROTOTYPE

The second prototype followed the design architecture created during the SE process. As a general concept, the second prototype served as a static example of the final product. A static configuration or “Config” page was created and populated with Strike data. The Config page can be seen in Figure 16.



Initial List of Functions For Each Phase

#	Primary Phase Function	#	Intermediate Phase Function	#	Advanced Phase Function	#	FRS Phase Function
1	Basic Instruments	1	Basic Instruments	1	Basic Instruments	1	A/C Handling
2	Radio Instuments	2	Radio Instuments	2	Radio Instuments	2	Instrument Procedures
3	Navigation	3	Navigation	3	Navigation	3	Emergency Procedures
4	Cockpit Procedures	4	Cockpit Procedures	4	Cockpit Procedures	4	Section Form
5	Emergency Procedures	5	Emergency Procedures	5	Emergency Procedures	5	Division Form
6	Airways Navigation	6	Airways Navigation	6	Airways Navigation	6	TACF
7	Low Level Navigation	7	Low Level Navigation	7	Low Level Navigation	7	Fwd QTR Intercepts
8	Instrument Rating	8	Instrument Rating	8	Instrument Rating	8	Low level Navigation
9	Operational Navigation	9	Operational Navigation	9	Operational Navigation	9	LATT
10	Familiarization	10	Familiarization	10	Familiarization	10	45/30 degree dives
11	Night Familiarization	11	Night Familiarization	11	Night Familiarization	11	Strafe
12	Formation	12	Formation	12	Formation	12	Pops
13	Low-level Formation	13	Low-level Formation	13	Low-level Formation	13	CAS
14	Tactical Formation	14	Tactical Formation	14	Tactical Formation	14	PGM
15	Night Formation	15	Night Formation	15	Night Formation	15	Sensor to Visual Handoff
16	FCLP	16	FCLP	16	FCLP	16	SACT
17	Carrier Qualification	17	Carrier Qualification	17	Carrier Qualification	17	Day IFR
18	Basic Fighter Maneuvering	18	Basic Fighter Maneuvering	18	Basic Fighter Maneuvering	18	Night IFR
19	Section Engaged Maneuvering	19	Section Engaged Maneuvering	19	Section Engaged Maneuvering	19	NVG
20	Strike	20	Strike	20	Strike	20	Perch BFM
						21	High Aspect BFM
						22	2v1 BFM
						23	FWD Qtr Tactics
						24	OCA
						25	DCA
						26	Day CQ
						27	Night CQ

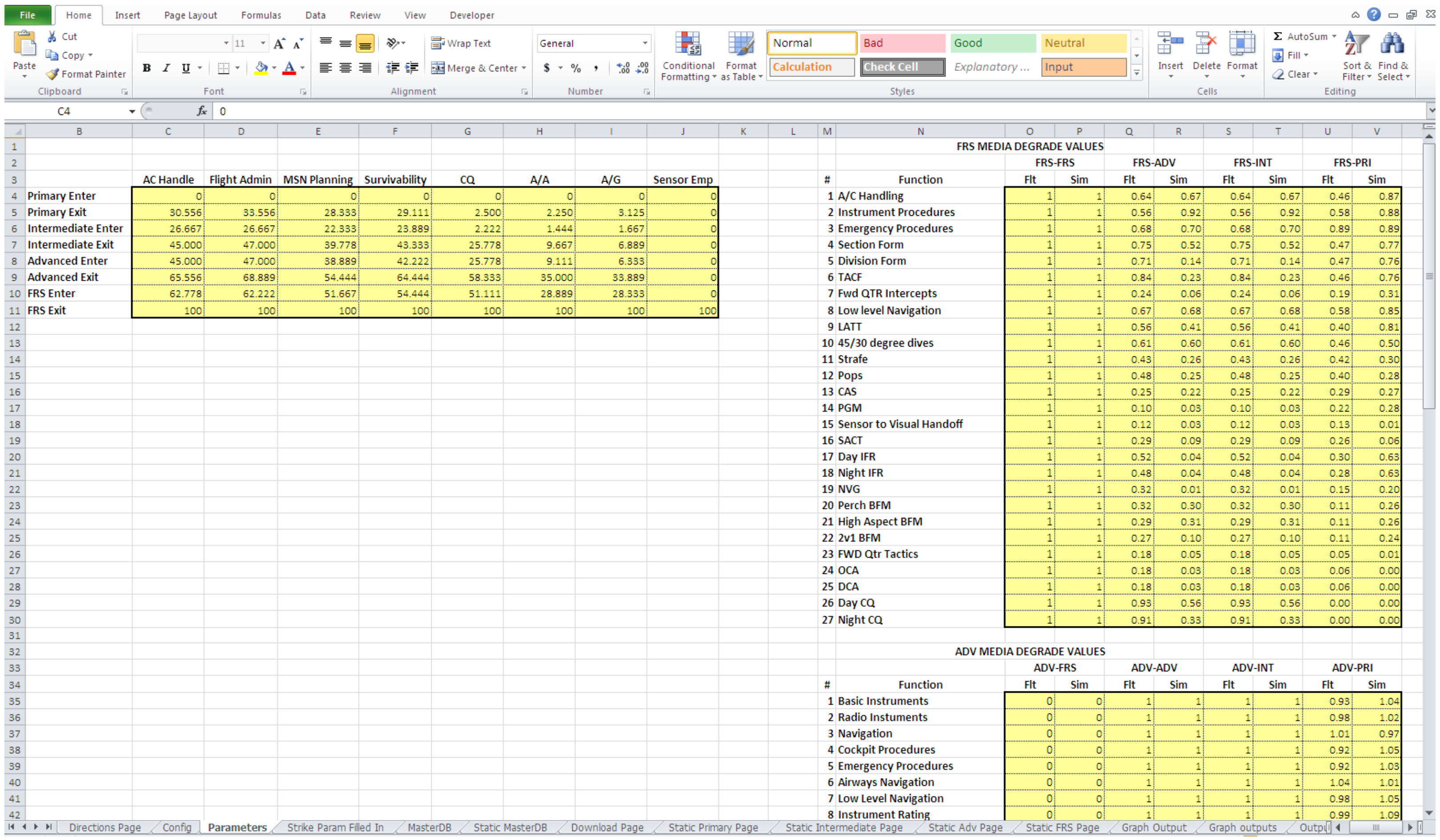
Pipeline Skill List
AC Handle
Flight Admin
MSN Planning
Survivability
CQ
A/A
A/G
Sensor Emp

Phase	Platform	Max Hr Factor	$ Flt	$ Sim
Primary	T-6B	2	$ 1,200	$ 500
Intermediate	T-45C	2	$ 3,924	$ 1,000
Advanced	T-45C	2	$ 3,924	$ 1,000
FRS	FA-18	1	$ 10,350	$ 1,500

Figure 16. Config Page For Second Prototype

The function lists for each phase as well as the pipeline skill list were hard coded into the model to minimize prototype automation.

Figure 17 shows a partial view of the static Parameters page. This page allows the user to input SME data; firewall proficiency values, and media degrader values.



	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Primary Enter	0	0	0	0	0	0	0	0
Primary Exit	30.556	33.556	28.333	29.111	2.500	2.250	3.125	0
Intermediate Enter	26.667	26.667	22.333	23.889	2.222	1.444	1.667	0
Intermediate Exit	45.000	47.000	39.778	43.333	25.778	9.667	6.889	0
Advanced Enter	45.000	47.000	38.889	42.222	25.778	9.111	6.333	0
Advanced Exit	65.556	68.889	54.444	64.444	58.333	35.000	33.889	0
FRS Enter	62.778	62.222	51.667	54.444	51.111	28.889	28.333	0
FRS Exit	100	100	100	100	100	100	100	100

FRS MEDIA DEGRADE VALUES

		FRS-FRS		FRS-ADV		FRS-INT		FRS-PRI	
#	Function	Flt	Sim	Flt	Sim	Flt	Sim	Flt	Sim
1	A/C Handling	1	1	0.64	0.67	0.64	0.67	0.46	0.87
2	Instrument Procedures	1	1	0.56	0.92	0.56	0.92	0.58	0.88
3	Emergency Procedures	1	1	0.68	0.70	0.68	0.70	0.89	0.89
4	Section Form	1	1	0.75	0.52	0.75	0.52	0.47	0.77
5	Division Form	1	1	0.71	0.14	0.71	0.14	0.47	0.76
6	TACF	1	1	0.84	0.23	0.84	0.23	0.46	0.76
7	Fwd QTR Intercepts	1	1	0.24	0.06	0.24	0.06	0.19	0.31
8	Low level Navigation	1	1	0.67	0.68	0.67	0.68	0.58	0.85
9	LATT	1	1	0.56	0.41	0.56	0.41	0.40	0.81
10	45/30 degree dives	1	1	0.61	0.60	0.61	0.60	0.46	0.50
11	Strafe	1	1	0.43	0.26	0.43	0.26	0.42	0.30
12	Pops	1	1	0.48	0.25	0.48	0.25	0.40	0.28
13	CAS	1	1	0.25	0.22	0.25	0.22	0.29	0.27
14	PGM	1	1	0.10	0.03	0.10	0.03	0.22	0.28
15	Sensor to Visual Handoff	1	1	0.12	0.03	0.12	0.03	0.13	0.01
16	SACT	1	1	0.29	0.09	0.29	0.09	0.26	0.06
17	Day IFR	1	1	0.52	0.04	0.52	0.04	0.30	0.63
18	Night IFR	1	1	0.48	0.04	0.48	0.04	0.28	0.63
19	NVG	1	1	0.32	0.01	0.32	0.01	0.15	0.20
20	Perch BFM	1	1	0.32	0.30	0.32	0.30	0.11	0.26
21	High Aspect BFM	1	1	0.29	0.31	0.29	0.31	0.11	0.26
22	2v1 BFM	1	1	0.27	0.10	0.27	0.10	0.11	0.24
23	FWD Qtr Tactics	1	1	0.18	0.05	0.18	0.05	0.05	0.01
24	OCA	1	1	0.18	0.03	0.18	0.03	0.06	0.00
25	DCA	1	1	0.18	0.03	0.18	0.03	0.06	0.00
26	Day CQ	1	1	0.93	0.56	0.93	0.56	0.00	0.00
27	Night CQ	1	1	0.91	0.33	0.91	0.33	0.00	0.00

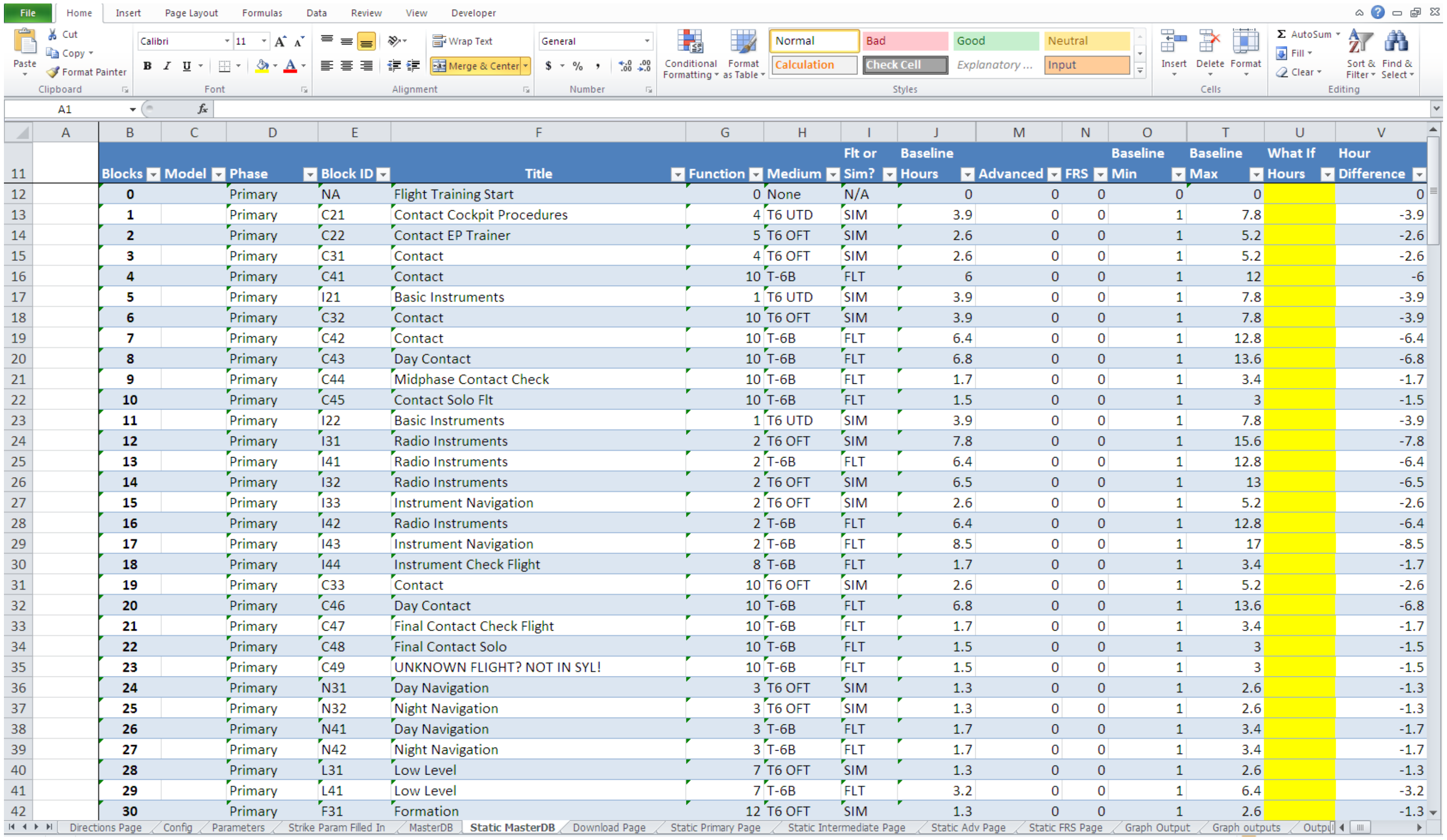
ADV MEDIA DEGRADE VALUES

		ADV-FRS		ADV-ADV		ADV-INT		ADV-PRI	
#	Function	Flt	Sim	Flt	Sim	Flt	Sim	Flt	Sim
1	Basic Instruments	0	0	1	1	1	1	0.93	1.04
2	Radio Instuments	0	0	1	1	1	1	0.98	1.02
3	Navigation	0	0	1	1	1	1	1.01	0.97
4	Cockpit Procedures	0	0	1	1	1	1	0.92	1.05
5	Emergency Procedures	0	0	1	1	1	1	0.92	1.03
6	Airways Navigation	0	0	1	1	1	1	1.04	1.01
7	Low Level Navigation	0	0	1	1	1	1	0.98	1.05
8	Instrument Rating	0	0	1	1	1	1	0.99	1.09

Figure 17. Parameters Page for Second Prototype (Partial Example)

Of the four training phases only three—FRS, Advanced, and Intermediate—are represented on the right of the page. Primary training was initially left out because downloading from Primary was not possible. In the final model, the Primary Phase was added back to allow future uploading analysis, should the need arise.

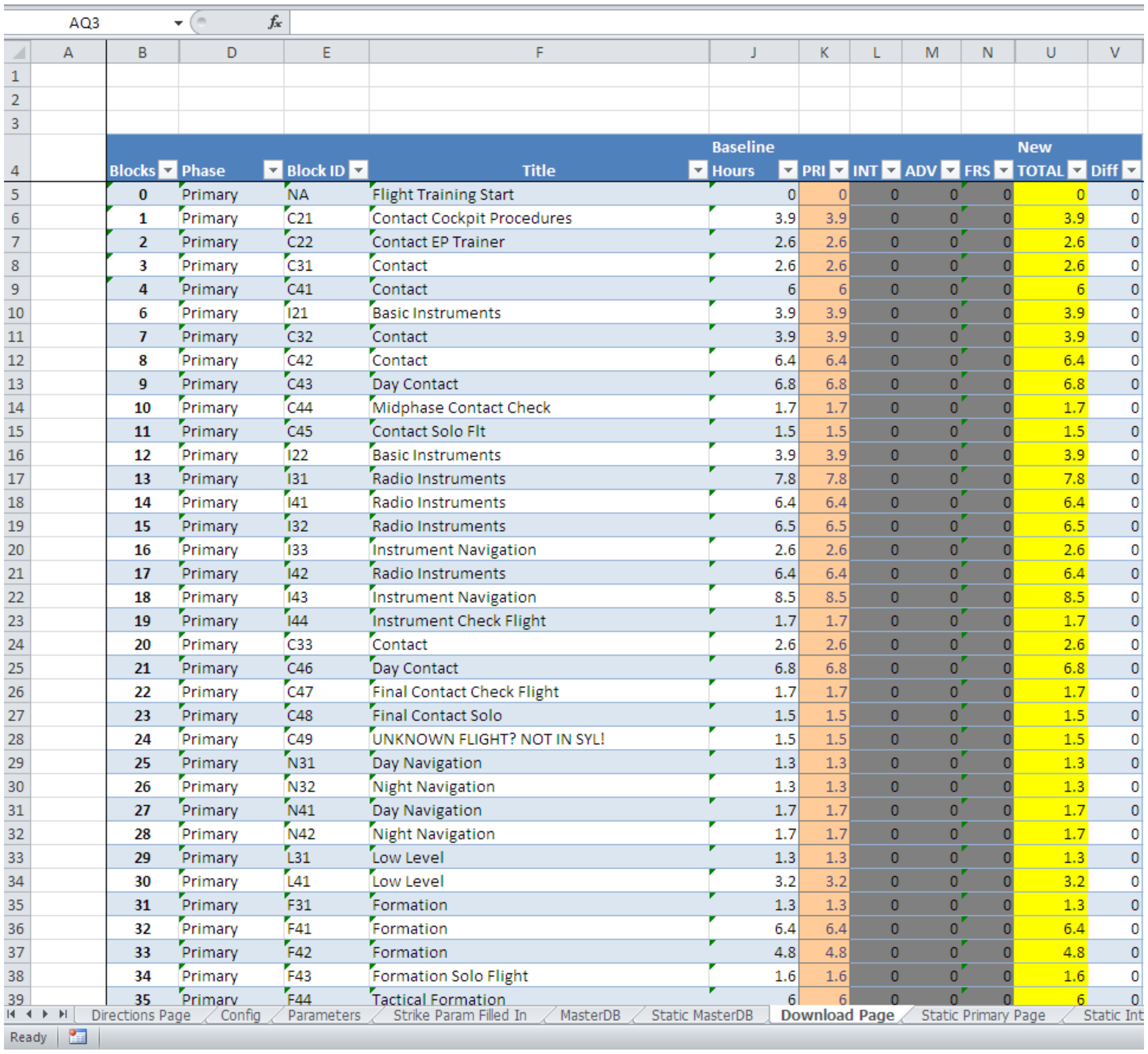
The Master DB page leveraged Excel Table functionality and was where all training pipeline blocks were entered. Once data is input in the correct format, it can be identified by various tags and sorted to help with analysis later. Figure 18 shows a partial picture of the Master DB in table format.



	A	B	C	D	E	F	G	H	I	J	M	N	O	T	U	V
11		Blocks	Model	Phase	Block ID	Title	Function	Medium	Flt or Sim?	Baseline Hours	Advanced	FRS	Baseline Min	Baseline Max	What If Hours	Hour Difference
12		0		Primary	NA	Flight Training Start	0	None	N/A	0	0	0	0	0		0
13		1		Primary	C21	Contact Cockpit Procedures	4	T6 UTD	SIM	3.9	0	0	1	7.8		-3.9
14		2		Primary	C22	Contact EP Trainer	5	T6 OFT	SIM	2.6	0	0	1	5.2		-2.6
15		3		Primary	C31	Contact	4	T6 OFT	SIM	2.6	0	0	1	5.2		-2.6
16		4		Primary	C41	Contact	10	T-6B	FLT	6	0	0	1	12		-6
17		5		Primary	I21	Basic Instruments	1	T6 UTD	SIM	3.9	0	0	1	7.8		-3.9
18		6		Primary	C32	Contact	10	T6 OFT	SIM	3.9	0	0	1	7.8		-3.9
19		7		Primary	C42	Contact	10	T-6B	FLT	6.4	0	0	1	12.8		-6.4
20		8		Primary	C43	Day Contact	10	T-6B	FLT	6.8	0	0	1	13.6		-6.8
21		9		Primary	C44	Midphase Contact Check	10	T-6B	FLT	1.7	0	0	1	3.4		-1.7
22		10		Primary	C45	Contact Solo Flt	10	T-6B	FLT	1.5	0	0	1	3		-1.5
23		11		Primary	I22	Basic Instruments	1	T6 UTD	SIM	3.9	0	0	1	7.8		-3.9
24		12		Primary	I31	Radio Instruments	2	T6 OFT	SIM	7.8	0	0	1	15.6		-7.8
25		13		Primary	I41	Radio Instruments	2	T-6B	FLT	6.4	0	0	1	12.8		-6.4
26		14		Primary	I32	Radio Instruments	2	T6 OFT	SIM	6.5	0	0	1	13		-6.5
27		15		Primary	I33	Instrument Navigation	2	T6 OFT	SIM	2.6	0	0	1	5.2		-2.6
28		16		Primary	I42	Radio Instruments	2	T-6B	FLT	6.4	0	0	1	12.8		-6.4
29		17		Primary	I43	Instrument Navigation	2	T-6B	FLT	8.5	0	0	1	17		-8.5
30		18		Primary	I44	Instrument Check Flight	8	T-6B	FLT	1.7	0	0	1	3.4		-1.7
31		19		Primary	C33	Contact	10	T6 OFT	SIM	2.6	0	0	1	5.2		-2.6
32		20		Primary	C46	Day Contact	10	T-6B	FLT	6.8	0	0	1	13.6		-6.8
33		21		Primary	C47	Final Contact Check Flight	10	T-6B	FLT	1.7	0	0	1	3.4		-1.7
34		22		Primary	C48	Final Contact Solo	10	T-6B	FLT	1.5	0	0	1	3		-1.5
35		23		Primary	C49	UNKNOWN FLIGHT? NOT IN SYL!	10	T-6B	FLT	1.5	0	0	1	3		-1.5
36		24		Primary	N31	Day Navigation	3	T6 OFT	SIM	1.3	0	0	1	2.6		-1.3
37		25		Primary	N32	Night Navigation	3	T6 OFT	SIM	1.3	0	0	1	2.6		-1.3
38		26		Primary	N41	Day Navigation	3	T-6B	FLT	1.7	0	0	1	3.4		-1.7
39		27		Primary	N42	Night Navigation	3	T-6B	FLT	1.7	0	0	1	3.4		-1.7
40		28		Primary	L31	Low Level	7	T6 OFT	SIM	1.3	0	0	1	2.6		-1.3
41		29		Primary	L41	Low Level	7	T-6B	FLT	3.2	0	0	1	6.4		-3.2
42		30		Primary	F31	Formation	12	T6 OFT	SIM	1.3	0	0	1	2.6		-1.3

Figure 18. Second Prototype MasterDB (Partial Example)

The Download page is where most of the user-defined scenario input takes place. This is where training hours can be moved or downloaded between platforms. The Download page is combined with the four static phase models to provide the final training pipeline model. Figure 19 shows a partial view of the Download page.



AQ3 fx

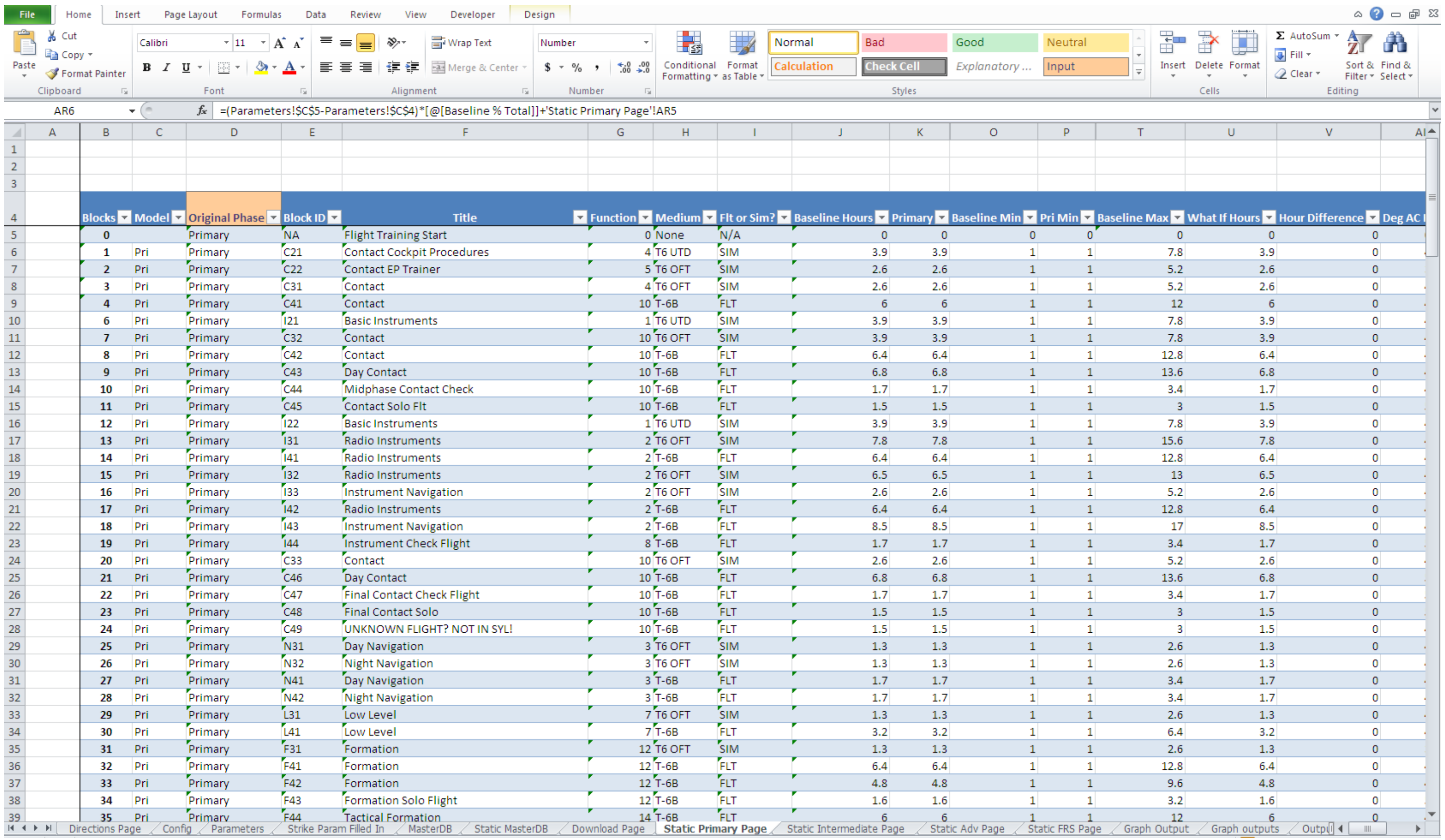
	A	B	D	E	F	J	K	L	M	N	U	V
1												
2												
3												
4		Blocks	Phase	Block ID	Title	Baseline Hours	PRI	INT	ADV	FRS	New TOTAL	Diff
5		0	Primary	NA	Flight Training Start	0	0	0	0	0	0	0
6		1	Primary	C21	Contact Cockpit Procedures	3.9	3.9	0	0	0	3.9	0
7		2	Primary	C22	Contact EP Trainer	2.6	2.6	0	0	0	2.6	0
8		3	Primary	C31	Contact	2.6	2.6	0	0	0	2.6	0
9		4	Primary	C41	Contact	6	6	0	0	0	6	0
10		6	Primary	I21	Basic Instruments	3.9	3.9	0	0	0	3.9	0
11		7	Primary	C32	Contact	3.9	3.9	0	0	0	3.9	0
12		8	Primary	C42	Contact	6.4	6.4	0	0	0	6.4	0
13		9	Primary	C43	Day Contact	6.8	6.8	0	0	0	6.8	0
14		10	Primary	C44	Midphase Contact Check	1.7	1.7	0	0	0	1.7	0
15		11	Primary	C45	Contact Solo Flt	1.5	1.5	0	0	0	1.5	0
16		12	Primary	I22	Basic Instruments	3.9	3.9	0	0	0	3.9	0
17		13	Primary	I31	Radio Instruments	7.8	7.8	0	0	0	7.8	0
18		14	Primary	I41	Radio Instruments	6.4	6.4	0	0	0	6.4	0
19		15	Primary	I32	Radio Instruments	6.5	6.5	0	0	0	6.5	0
20		16	Primary	I33	Instrument Navigation	2.6	2.6	0	0	0	2.6	0
21		17	Primary	I42	Radio Instruments	6.4	6.4	0	0	0	6.4	0
22		18	Primary	I43	Instrument Navigation	8.5	8.5	0	0	0	8.5	0
23		19	Primary	I44	Instrument Check Flight	1.7	1.7	0	0	0	1.7	0
24		20	Primary	C33	Contact	2.6	2.6	0	0	0	2.6	0
25		21	Primary	C46	Day Contact	6.8	6.8	0	0	0	6.8	0
26		22	Primary	C47	Final Contact Check Flight	1.7	1.7	0	0	0	1.7	0
27		23	Primary	C48	Final Contact Solo	1.5	1.5	0	0	0	1.5	0
28		24	Primary	C49	UNKNOWN FLIGHT? NOT IN SYL!	1.5	1.5	0	0	0	1.5	0
29		25	Primary	N31	Day Navigation	1.3	1.3	0	0	0	1.3	0
30		26	Primary	N32	Night Navigation	1.3	1.3	0	0	0	1.3	0
31		27	Primary	N41	Day Navigation	1.7	1.7	0	0	0	1.7	0
32		28	Primary	N42	Night Navigation	1.7	1.7	0	0	0	1.7	0
33		29	Primary	L31	Low Level	1.3	1.3	0	0	0	1.3	0
34		30	Primary	L41	Low Level	3.2	3.2	0	0	0	3.2	0
35		31	Primary	F31	Formation	1.3	1.3	0	0	0	1.3	0
36		32	Primary	F41	Formation	6.4	6.4	0	0	0	6.4	0
37		33	Primary	F42	Formation	4.8	4.8	0	0	0	4.8	0
38		34	Primary	F43	Formation Solo Flight	1.6	1.6	0	0	0	1.6	0
39		35	Primary	F44	Tactical Formation	6	6	0	0	0	6	0

Directions Page / Config / Parameters / Strike Param Filled In / MasterDB / Static MasterDB / Download Page / Static Primary Page / Static Int

Ready

Figure 19. Second Prototype Download Page (Partial Example)

Static phase models were also created to make sure the layout supported automation. An example of a phase model is captured in Figure 20.



AR6 =(Parameters!$C$5-Parameters!$C$4)*[@[Baseline % Total]]+'Static Primary Page'!AR5

Blocks	Model	Original Phase	Block ID	Title	Function	Medium	Flt or Sim?	Baseline Hours	Primary	Baseline Min	Pri Min	Baseline Max	What If Hours	Hour Difference	Deg AC
0		Primary	NA	Flight Training Start	0	None	N/A	0	0	0	0	0	0	0	
1	Pri	Primary	C21	Contact Cockpit Procedures	4	T6 UTD	SIM	3.9	3.9	1	1	7.8	3.9	0	
2	Pri	Primary	C22	Contact EP Trainer	5	T6 OFT	SIM	2.6	2.6	1	1	5.2	2.6	0	
3	Pri	Primary	C31	Contact	4	T6 OFT	SIM	2.6	2.6	1	1	5.2	2.6	0	
4	Pri	Primary	C41	Contact	10	T-6B	FLT	6	6	1	1	12	6	0	
6	Pri	Primary	I21	Basic Instruments	1	T6 UTD	SIM	3.9	3.9	1	1	7.8	3.9	0	
7	Pri	Primary	C32	Contact	10	T6 OFT	SIM	3.9	3.9	1	1	7.8	3.9	0	
8	Pri	Primary	C42	Contact	10	T-6B	FLT	6.4	6.4	1	1	12.8	6.4	0	
9	Pri	Primary	C43	Day Contact	10	T-6B	FLT	6.8	6.8	1	1	13.6	6.8	0	
10	Pri	Primary	C44	Midphase Contact Check	10	T-6B	FLT	1.7	1.7	1	1	3.4	1.7	0	
11	Pri	Primary	C45	Contact Solo Flt	10	T-6B	FLT	1.5	1.5	1	1	3	1.5	0	
12	Pri	Primary	I22	Basic Instruments	1	T6 UTD	SIM	3.9	3.9	1	1	7.8	3.9	0	
13	Pri	Primary	I31	Radio Instruments	2	T6 OFT	SIM	7.8	7.8	1	1	15.6	7.8	0	
14	Pri	Primary	I41	Radio Instruments	2	T-6B	FLT	6.4	6.4	1	1	12.8	6.4	0	
15	Pri	Primary	I32	Radio Instruments	2	T6 OFT	SIM	6.5	6.5	1	1	13	6.5	0	
16	Pri	Primary	I33	Instrument Navigation	2	T6 OFT	SIM	2.6	2.6	1	1	5.2	2.6	0	
17	Pri	Primary	I42	Radio Instruments	2	T-6B	FLT	6.4	6.4	1	1	12.8	6.4	0	
18	Pri	Primary	I43	Instrument Navigation	2	T-6B	FLT	8.5	8.5	1	1	17	8.5	0	
19	Pri	Primary	I44	Instrument Check Flight	8	T-6B	FLT	1.7	1.7	1	1	3.4	1.7	0	
20	Pri	Primary	C33	Contact	10	T6 OFT	SIM	2.6	2.6	1	1	5.2	2.6	0	
21	Pri	Primary	C46	Day Contact	10	T-6B	FLT	6.8	6.8	1	1	13.6	6.8	0	
22	Pri	Primary	C47	Final Contact Check Flight	10	T-6B	FLT	1.7	1.7	1	1	3.4	1.7	0	
23	Pri	Primary	C48	Final Contact Solo	10	T-6B	FLT	1.5	1.5	1	1	3	1.5	0	
24	Pri	Primary	C49	UNKNOWN FLIGHT? NOT IN SYL!	10	T-6B	FLT	1.5	1.5	1	1	3	1.5	0	
25	Pri	Primary	N31	Day Navigation	3	T6 OFT	SIM	1.3	1.3	1	1	2.6	1.3	0	
26	Pri	Primary	N32	Night Navigation	3	T6 OFT	SIM	1.3	1.3	1	1	2.6	1.3	0	
27	Pri	Primary	N41	Day Navigation	3	T-6B	FLT	1.7	1.7	1	1	3.4	1.7	0	
28	Pri	Primary	N42	Night Navigation	3	T-6B	FLT	1.7	1.7	1	1	3.4	1.7	0	
29	Pri	Primary	L31	Low Level	7	T6 OFT	SIM	1.3	1.3	1	1	2.6	1.3	0	
30	Pri	Primary	L41	Low Level	7	T-6B	FLT	3.2	3.2	1	1	6.4	3.2	0	
31	Pri	Primary	F31	Formation	12	T6 OFT	SIM	1.3	1.3	1	1	2.6	1.3	0	
32	Pri	Primary	F41	Formation	12	T-6B	FLT	6.4	6.4	1	1	12.8	6.4	0	
33	Pri	Primary	F42	Formation	12	T-6B	FLT	4.8	4.8	1	1	9.6	4.8	0	
34	Pri	Primary	F43	Formation Solo Flight	12	T-6B	FLT	1.6	1.6	1	1	3.2	1.6	0	
35	Pri	Primary	F44	Tactical Formation	14	T-6B	FLT	6	6	1	1	12	6	0	

Figure 20. Second Prototype Phase Model (Partial Example)

These models were created as Excel tables to take advantage of the inherent functionality of the table format. This translated directly into the final model as discoveries were made during this process that simplified the coding process significantly. Specifically, table functionality allows entire columns to be filled based on the contents of the first cell. As a result, final coding only had to input formulas into the top cell and it would automatically fill the rest.

Graphical outputs of the second prototype were not optimal. At this point, the design of the final graphical output was still in question. Developing an output that provided the information clearly and concisely to the user required multiple attempts. Figure 21 shows the output of the second prototype.

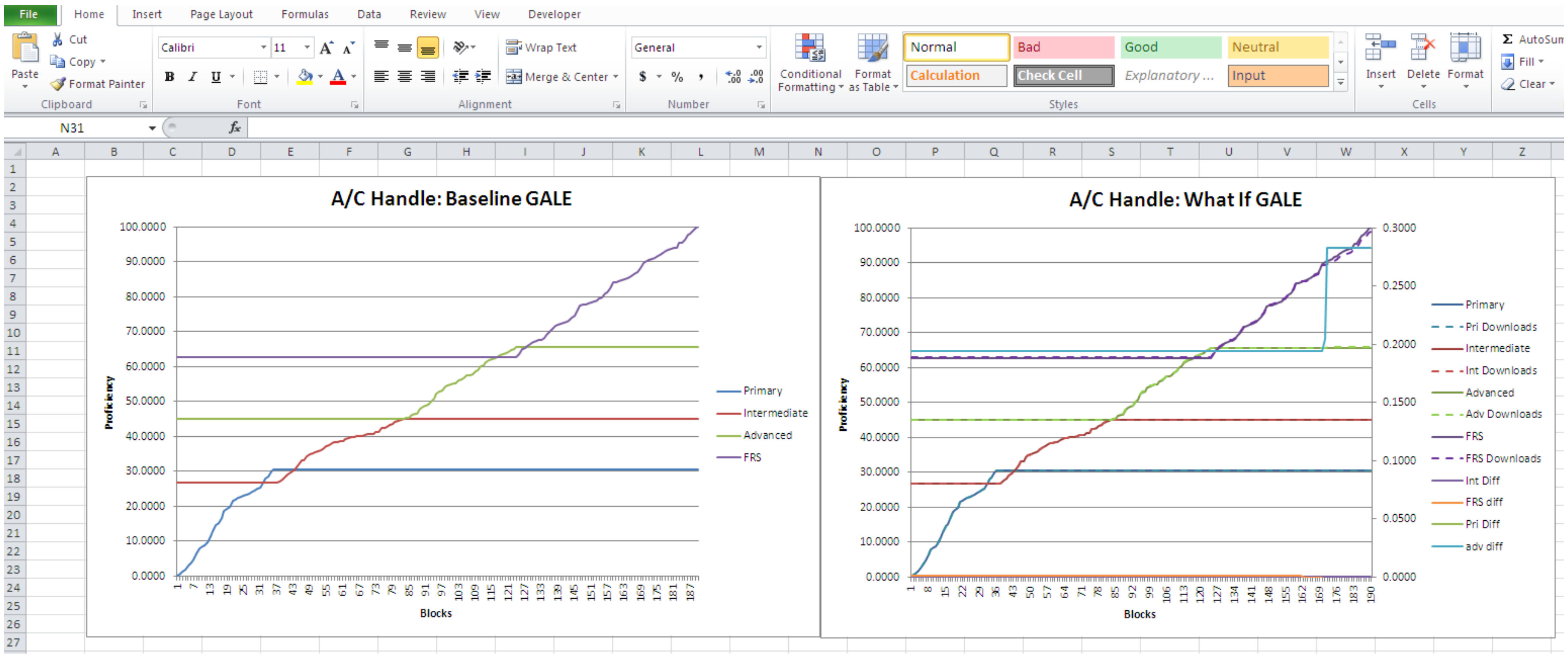


Figure 21. Skill Proficiency Output of Second Prototype

The horizontal lines provided Phase entry and exit levels. The general shape of the learning curve showed proficiency gain. This output proved to be undesirable. Although the information was displayed, it was difficult to interpret and understand. Looking at what was truly important to see—the entry and exit points of the phases as well as the final proficiency in each skill—a new display for model outputs was designed into the final model.

APPENDIX D: FULL DESCRIPTION OF NAVAL AVIATION PROFICIENCY ANALYSIS MODEL

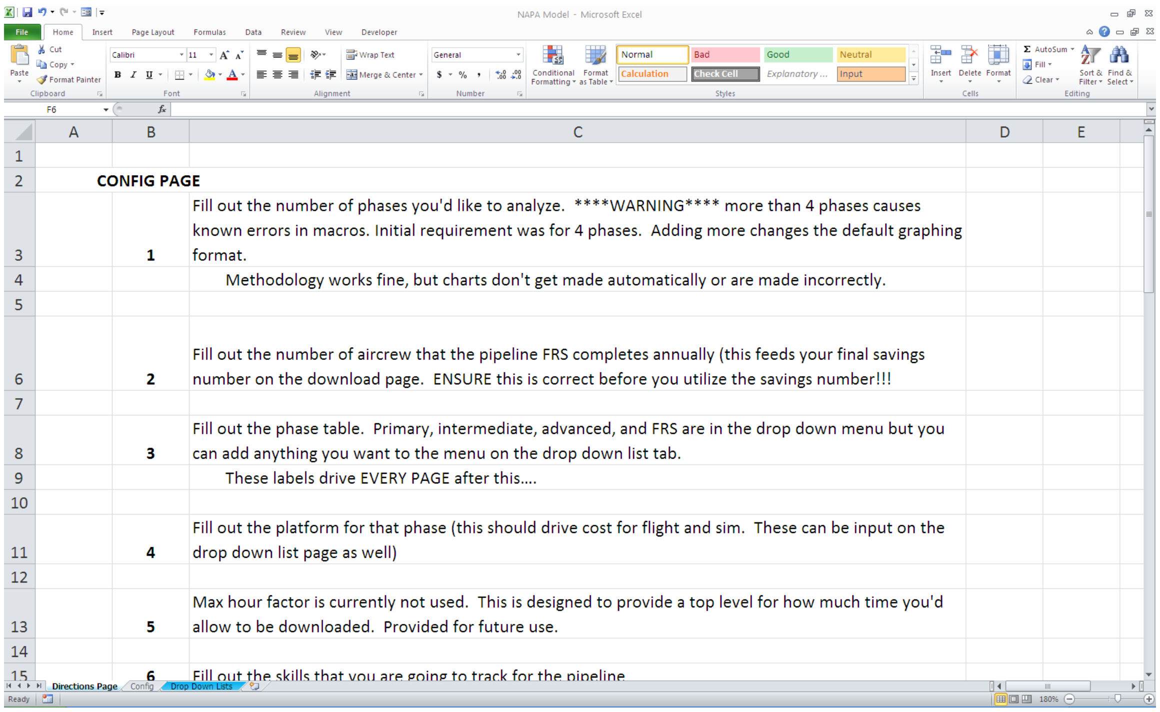
An in-depth look into the final design of the Naval Aviation Proficiency Analysis (NAPA) model provides the insight necessary to ensure the system behaviors and requirements are met. The interfaces between the pages were facilitated primarily by VBA code. Appendix C contains the entire applications code for review.

A. FINAL PAGE DESIGNS

When users want to create a new NAPA model, they open the master file and find three pages; the Directions page, CONFIG page, and the Dropdown Menu Page. Equipped with training pipeline specifics, users systematically fill out the model and ultimately create a live worksheet capable of facilitating exploration of downloading events from one phase to another. In general, user-input cells are highlighted in yellow throughout most of the model for human interface considerations. Every page in the model has functionality that, together, provides the user with a simple way to analyze efficiency excursions. Detailed inspection of the final page designs accomplishes three things: it helps depict the model's capabilities; it delineates the physical allocation of each requirement to a component of the model; and it describes the contribution each component has to the overall system behaviors required of the model.

1. Pages Available When Creating A New Model

The **Directions page** is designed as a quick reference for the user to understand what was required to make the model work. Basic user-interface information is supplied to help guide a user to creating a working model. This page helps provide a clear guide for user interface with the model. Additionally, it provides an easily accessible "help" function for the user. Once the user becomes familiar with the model, little interaction with this page would occur. An example of the Directions page is provided in Figure 22.

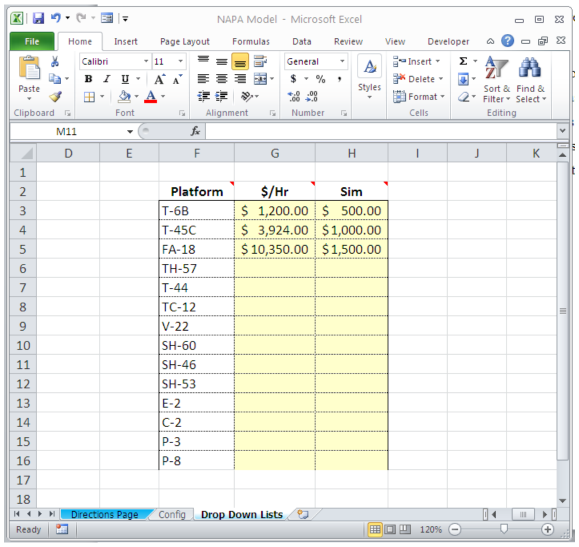


	A	B	C	D	E
1					
2		**CONFIG PAGE**			
3		**1**	Fill out the number of phases you'd like to analyze. ****WARNING**** more than 4 phases causes known errors in macros. Initial requirement was for 4 phases. Adding more changes the default graphing format.		
4			Methodology works fine, but charts don't get made automatically or are made incorrectly.		
5					
6		**2**	Fill out the number of aircrew that the pipeline FRS completes annually (this feeds your final savings number on the download page. ENSURE this is correct before you utilize the savings number!!!		
7					
8		**3**	Fill out the phase table. Primary, intermediate, advanced, and FRS are in the drop down menu but you can add anything you want to the menu on the drop down list tab.		
9			These labels drive EVERY PAGE after this....		
10					
11		**4**	Fill out the platform for that phase (this should drive cost for flight and sim. These can be input on the drop down list page as well)		
12					
13		**5**	Max hour factor is currently not used. This is designed to provide a top level for how much time you'd allow to be downloaded. Provided for future use.		
14					
15		**6**	Fill out the skills that you are going to track for the pipeline		

Figure 22. NAPA Model Directions Page

Figure 22 only shows a portion of the directions provided on the page.

The **Drop Down List page** (Figure 23) allows the user to input the cost per hour of operating different platforms and simulators. Since the Strike pipeline was the test pipeline for this effort, only Strike platform costs are shown.



Platform	$/Hr	Sim
T-6B	$ 1,200.00	$ 500.00
T-45C	$ 3,924.00	$1,000.00
FA-18	$ 10,350.00	$1,500.00
TH-57		
T-44		
TC-12		
V-22		
SH-60		
SH-46		
SH-53		
E-2		
C-2		
P-3		
P-8		

Figure 23. Drop Down List Page

To make the model extensible, this list had to be dynamic, to properly populate the rest of the model. If a platform was not listed it could be added without further modification to the model for utilization. Additionally, to keep the model up-to-date, the Drop Down List must be updated with the most recent operational costs of the platforms and simulators to maintain model validity.

The **CONFIG page** (Figures 24 and 25) is where the model becomes fundamentally extensible. It contains a few crucial components that require user-definition for the model to properly be created. Specifically:

The total number of phases must be defined. Initially, four phases were hard coded into the design based on an early assumption to scope the problem. However, providing a model with the capability to analyze more than four phases enhances the extensibility of the model and allows it to be utilized for future analysis efforts where follow-on Fleet phases of the training continuum may be added. Should greater than four phases of training ever be implemented, the graphical output of the model would have to be manually altered as the default graph changes.

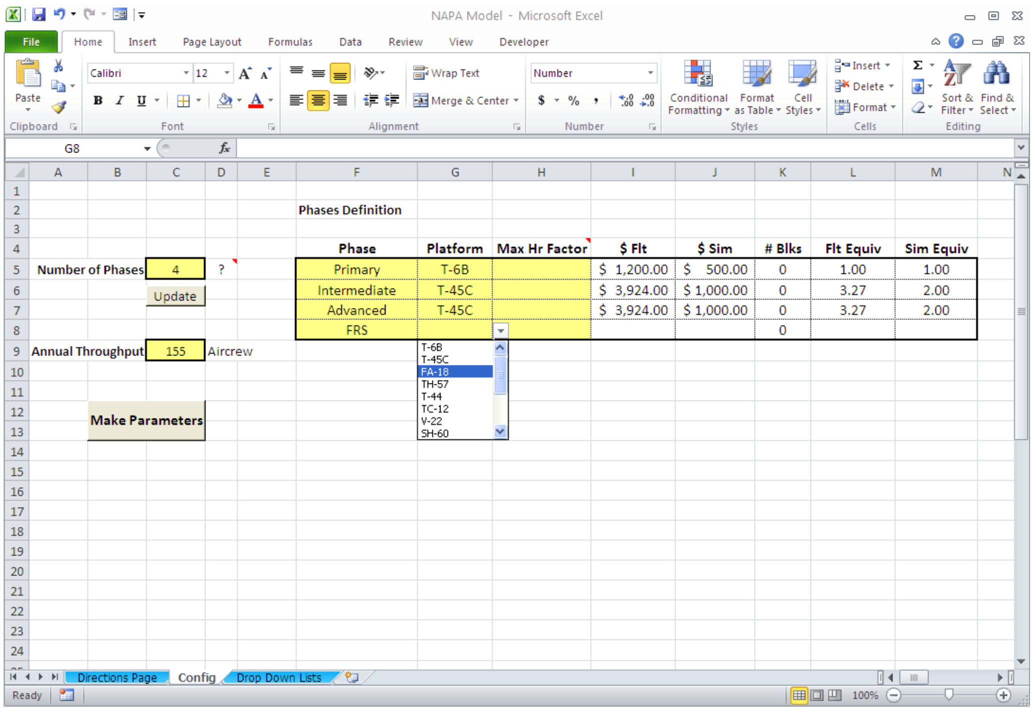
Phase names must be defined. This is where the model derives subsequent labels. Figure 24 shows the four common phase names defined although any user-defined names could be used and would propagate throughout the model.

Annual throughput for the pipeline must be defined. This is where the model looks to assist in the cost analysis portion of the model. Improper identification of this throughput would propagate inaccuracies throughout the cost analysis of the model.

Pipeline Skills and threshold proficiency levels must be identified. Similar to the phase names, the user-defined list of skills is where the model looks to properly configure itself. Figure 16 shows a completed Skill list and proficiency threshold for the Strike pipeline.

Phase functions must be identified. This is where the model begins to build search capability for its media degrader values.

Figure 24 depicts what the user would see immediately upon opening the page.



Phases Definition

Number of Phases: 4 ?

Update

Annual Throughput: 155 Aircrew

Make Parameters

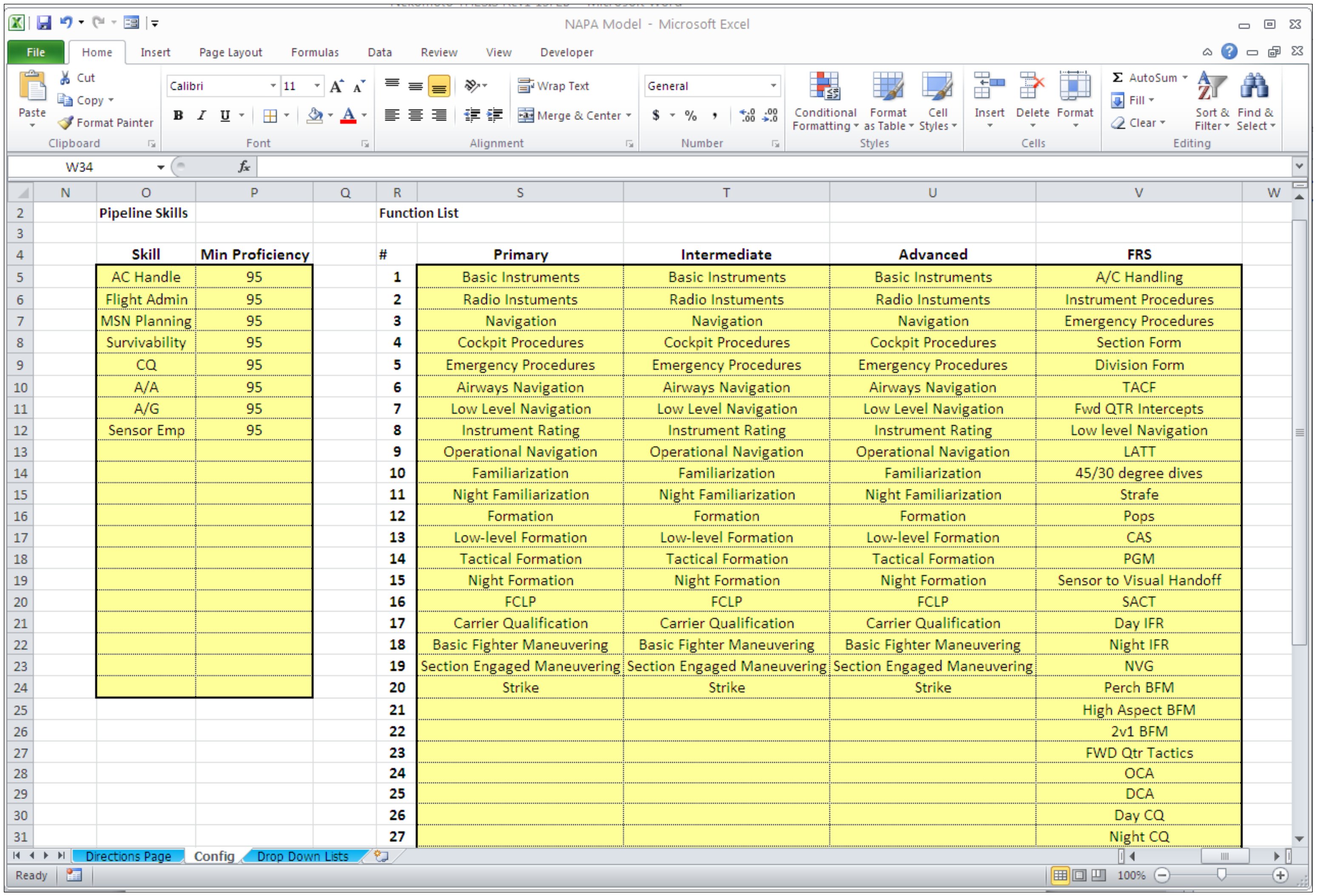
Phase	Platform	Max Hr Factor	$ Flt	$ Sim	# Blks	Flt Equiv	Sim Equiv
Primary	T-6B		$ 1,200.00	$ 500.00	0	1.00	1.00
Intermediate	T-45C		$ 3,924.00	$ 1,000.00	0	3.27	2.00
Advanced	T-45C		$ 3,924.00	$ 1,000.00	0	3.27	2.00
FRS					0		

T-6B
T-45C
FA-18
TH-57
T-44
TC-12
V-22
SH-60

Figure 24. CONFIG Page Part I: Phases, Platforms, and Throughput

The interaction between the Drop Down List and the CONFIG page is captured in Figure 24, as user defined Platforms show up for selection in the platform column.

Once completed, the user scrolls to the right to reveal the rest of the page, shown in Figure 25. This is where the rest of the user-defined input on the CONFIG page is input.



Pipeline Skills

Skill	Min Proficiency
AC Handle	95
Flight Admin	95
MSN Planning	95
Survivability	95
CQ	95
A/A	95
A/G	95
Sensor Emp	95

Function List

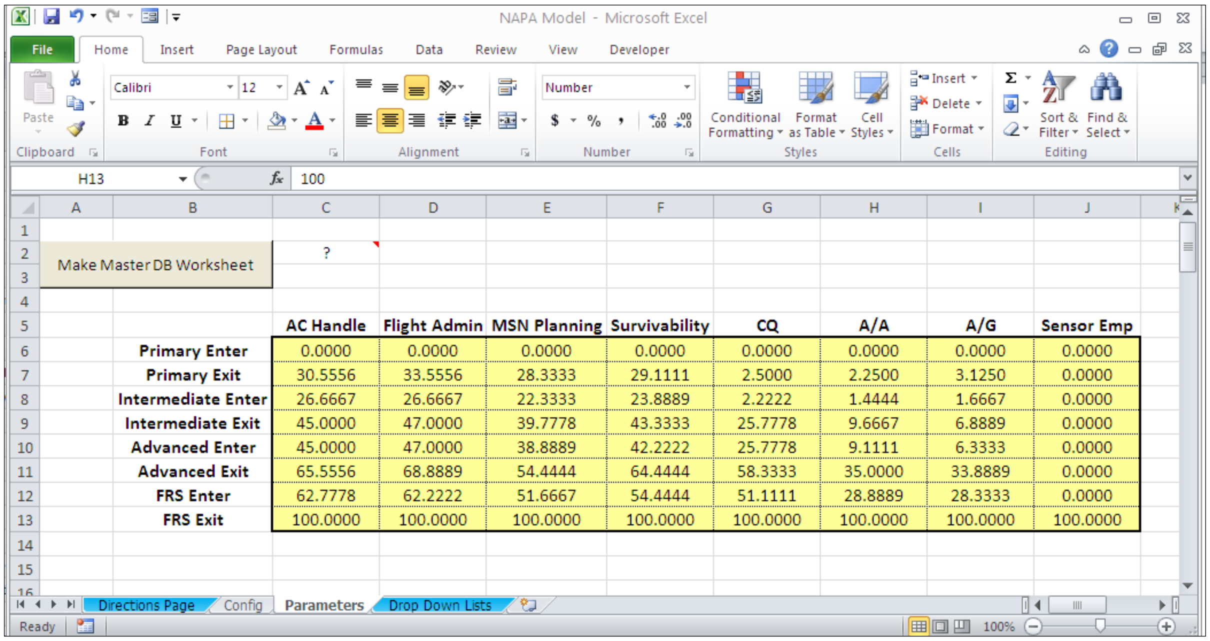
#	Primary	Intermediate	Advanced	FRS
1	Basic Instruments	Basic Instruments	Basic Instruments	A/C Handling
2	Radio Instuments	Radio Instuments	Radio Instuments	Instrument Procedures
3	Navigation	Navigation	Navigation	Emergency Procedures
4	Cockpit Procedures	Cockpit Procedures	Cockpit Procedures	Section Form
5	Emergency Procedures	Emergency Procedures	Emergency Procedures	Division Form
6	Airways Navigation	Airways Navigation	Airways Navigation	TACF
7	Low Level Navigation	Low Level Navigation	Low Level Navigation	Fwd QTR Intercepts
8	Instrument Rating	Instrument Rating	Instrument Rating	Low level Navigation
9	Operational Navigation	Operational Navigation	Operational Navigation	LATT
10	Familiarization	Familiarization	Familiarization	45/30 degree dives
11	Night Familiarization	Night Familiarization	Night Familiarization	Strafe
12	Formation	Formation	Formation	Pops
13	Low-level Formation	Low-level Formation	Low-level Formation	CAS
14	Tactical Formation	Tactical Formation	Tactical Formation	PGM
15	Night Formation	Night Formation	Night Formation	Sensor to Visual Handoff
16	FCLP	FCLP	FCLP	SACT
17	Carrier Qualification	Carrier Qualification	Carrier Qualification	Day IFR
18	Basic Fighter Maneuvering	Basic Fighter Maneuvering	Basic Fighter Maneuvering	Night IFR
19	Section Engaged Maneuvering	Section Engaged Maneuvering	Section Engaged Maneuvering	NVG
20	Strike	Strike	Strike	Perch BFM
21				High Aspect BFM
22				2v1 BFM
23				FWD Qtr Tactics
24				OCA
25				DCA
26				Day CQ
27				Night CQ

Figure 25. CONFIG Page Part II: Skills, Proficiency Thresholds, and Phase Functions

Once completed, a subroutine called “MakeParametersPage()” is called by depressing the “Make Parameters” button on the CONFIG page shown in Figure 24.

2. The Parameters Page

The **Parameters page** relies on CONFIG page inputs to be properly created. The Skill list is required to create the properly sized baseline proficiency or “firewall” table shown in Figure 26.

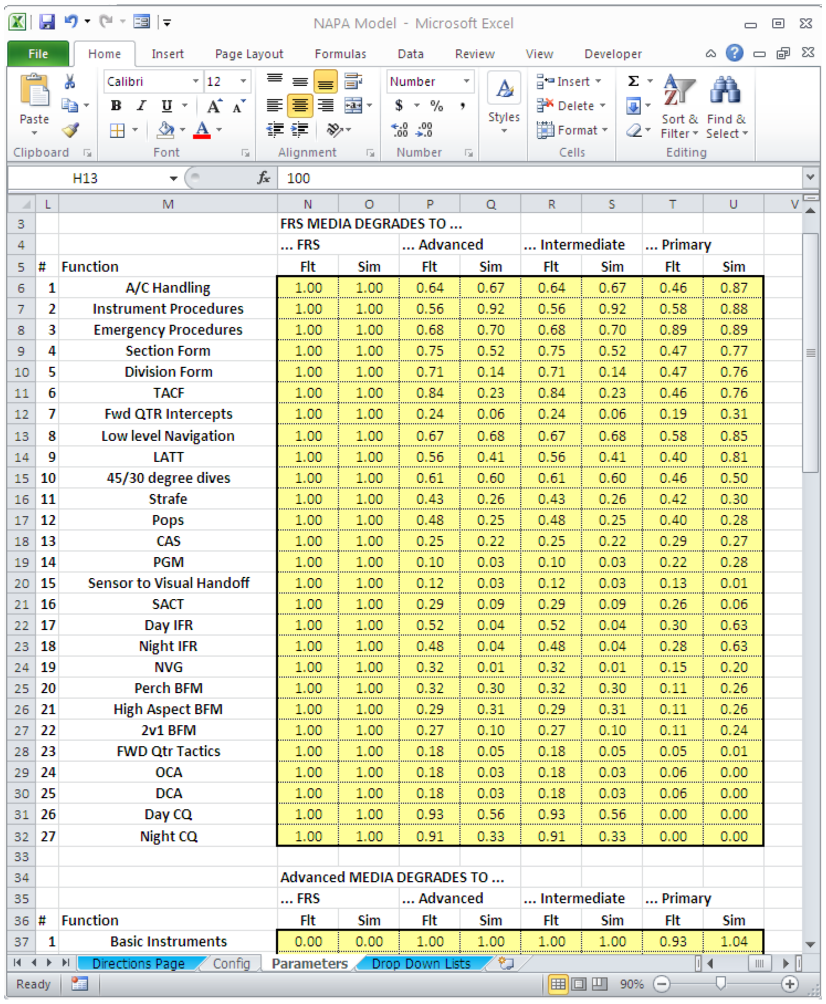


	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Primary Enter	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Primary Exit	30.5556	33.5556	28.3333	29.1111	2.5000	2.2500	3.1250	0.0000
Intermediate Enter	26.6667	26.6667	22.3333	23.8889	2.2222	1.4444	1.6667	0.0000
Intermediate Exit	45.0000	47.0000	39.7778	43.3333	25.7778	9.6667	6.8889	0.0000
Advanced Enter	45.0000	47.0000	38.8889	42.2222	25.7778	9.1111	6.3333	0.0000
Advanced Exit	65.5556	68.8889	54.4444	64.4444	58.3333	35.0000	33.8889	0.0000
FRS Enter	62.7778	62.2222	51.6667	54.4444	51.1111	28.8889	28.3333	0.0000
FRS Exit	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Figure 26. Parameters Page Part I: Baseline Proficiency Table

The table also relies on the user-defined Phase names for its row labels. This table is where the baseline phase proficiency values provided by SME input would be captured.

Once the baseline proficiency value table is filled out, the user has to fill out the media degrader tables on the page. This is where SME provided platform proficiency degrader values are tagged with the function they affect. Figure 27 shows an example of one of the four media degrader tables for the Strike syllabus.



H13 | 100

		FRS MEDIA DEGRADES TO ...							
		... FRS		... Advanced		... Intermediate		... Primary	
#	Function	Flt	Sim	Flt	Sim	Flt	Sim	Flt	Sim
1	A/C Handling	1.00	1.00	0.64	0.67	0.64	0.67	0.46	0.87
2	Instrument Procedures	1.00	1.00	0.56	0.92	0.56	0.92	0.58	0.88
3	Emergency Procedures	1.00	1.00	0.68	0.70	0.68	0.70	0.89	0.89
4	Section Form	1.00	1.00	0.75	0.52	0.75	0.52	0.47	0.77
5	Division Form	1.00	1.00	0.71	0.14	0.71	0.14	0.47	0.76
6	TACF	1.00	1.00	0.84	0.23	0.84	0.23	0.46	0.76
7	Fwd QTR Intercepts	1.00	1.00	0.24	0.06	0.24	0.06	0.19	0.31
8	Low level Navigation	1.00	1.00	0.67	0.68	0.67	0.68	0.58	0.85
9	LATT	1.00	1.00	0.56	0.41	0.56	0.41	0.40	0.81
10	45/30 degree dives	1.00	1.00	0.61	0.60	0.61	0.60	0.46	0.50
11	Strafe	1.00	1.00	0.43	0.26	0.43	0.26	0.42	0.30
12	Pops	1.00	1.00	0.48	0.25	0.48	0.25	0.40	0.28
13	CAS	1.00	1.00	0.25	0.22	0.25	0.22	0.29	0.27
14	PGM	1.00	1.00	0.10	0.03	0.10	0.03	0.22	0.28
15	Sensor to Visual Handoff	1.00	1.00	0.12	0.03	0.12	0.03	0.13	0.01
16	SACT	1.00	1.00	0.29	0.09	0.29	0.09	0.26	0.06
17	Day IFR	1.00	1.00	0.52	0.04	0.52	0.04	0.30	0.63
18	Night IFR	1.00	1.00	0.48	0.04	0.48	0.04	0.28	0.63
19	NVG	1.00	1.00	0.32	0.01	0.32	0.01	0.15	0.20
20	Perch BFM	1.00	1.00	0.32	0.30	0.32	0.30	0.11	0.26
21	High Aspect BFM	1.00	1.00	0.29	0.31	0.29	0.31	0.11	0.26
22	2v1 BFM	1.00	1.00	0.27	0.10	0.27	0.10	0.11	0.24
23	FWD Qtr Tactics	1.00	1.00	0.18	0.05	0.18	0.05	0.05	0.01
24	OCA	1.00	1.00	0.18	0.03	0.18	0.03	0.06	0.00
25	DCA	1.00	1.00	0.18	0.03	0.18	0.03	0.06	0.00
26	Day CQ	1.00	1.00	0.93	0.56	0.93	0.56	0.00	0.00
27	Night CQ	1.00	1.00	0.91	0.33	0.91	0.33	0.00	0.00

		Advanced MEDIA DEGRADES TO ...							
		... FRS		... Advanced		... Intermediate		... Primary	
#	Function	Flt	Sim	Flt	Sim	Flt	Sim	Flt	Sim
1	Basic Instruments	0.00	0.00	1.00	1.00	1.00	1.00	0.93	1.04

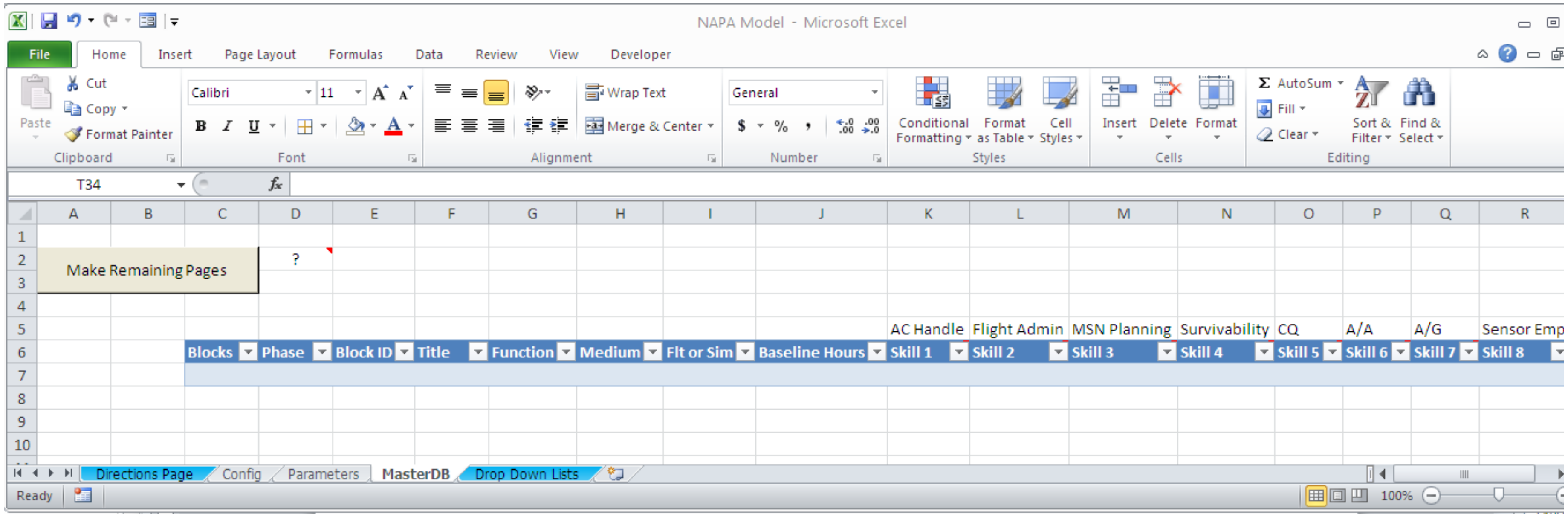
Figure 27. Parameters Page Part II: Sample (1 of 4) Media Degrader Tables

The media degrader tables are designed to support the potential for future "uploading" analysis as well. However, due to the fact that there is no data on uploading, the columns are effectively negated by zeroing out the column—essentially saying that there is NO value provided for moving a flight to a higher cost platform. When uploading data becomes available it can be quickly incorporated into the model for analysis.

Upon completion, the MasterDB page is created by calling a subroutine titled "MakeMasterDBPage()" by pressing the appropriate user-interface button located on the page.

3. The MasterDB Page

The **MasterDB page** is where the entire block structure of the pipeline is input. Additionally, this is where the Mean Effectiveness Ratings or MERs are first input into the model. An example of a blank MasterDB page is shown in Figure 28.



								AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Blocks	Phase	Block ID	Title	Function	Medium	Flt or Sim	Baseline Hours	Skill 1	Skill 2	Skill 3	Skill 4	Skill 5	Skill 6	Skill 7	Skill 8

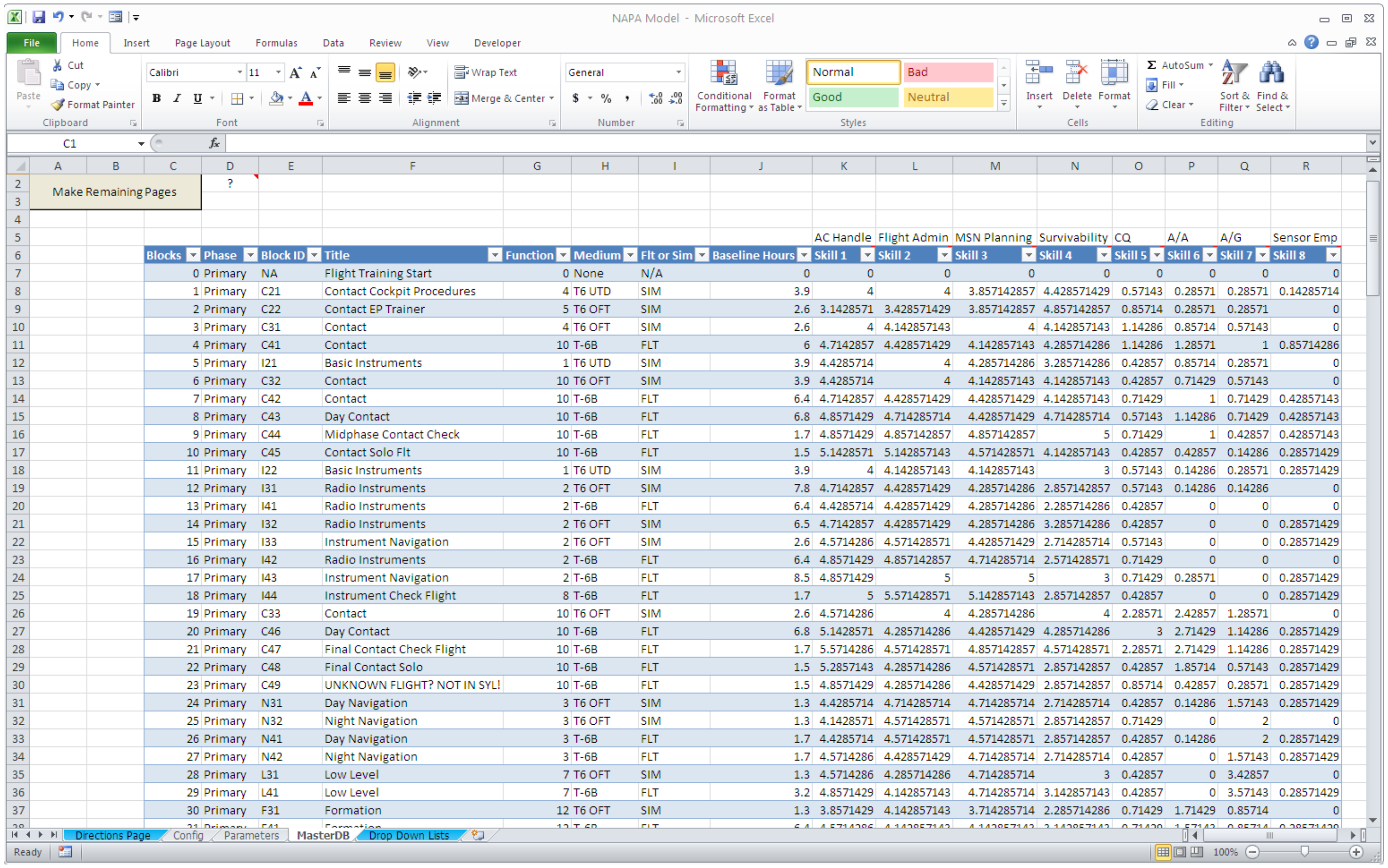
Figure 28. Initial MasterDB Page

This is the first time that a deviation from the yellow-highlighted cell for user input appears. Excel's inherent table properties satisfy user needs and create system behaviors required to satisfy the requirements. After consulting with the users, the table format was accepted.

Data for the Master DB page must contain the following values for the model to accept it and provide proper results:

- The original Phase in which the block is completed
- The Block ID or alpha-numeric block-identifier that ties directly to the phase syllabus. (Usually SME provided)
- Title of the Block appears next to the Block ID for plain English identification
- Phase function that the Block services is indicated
- The platform in which each Block is completed
- A block tag to identify it as a flight block or a simulator block (to help identify blocks for cost analysis purposes)
- The syllabus baseline hours for each block is noted
- The Mean Effectiveness Rating (MER) value that the block provides for each skill identified in the pipeline skill list

Filled out, the MasterDB becomes a sortable table that provides the mapping between block, platform, MER value, function, Media Degrader Value, and Skill. An example of the first 30 blocks of the Strike pipeline is shown in Figure 29.



NAPA Model - Microsoft Excel

Make Remaining Pages ?

								AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Blocks	Phase	Block ID	Title	Function	Medium	Flt or Sim	Baseline Hours	Skill 1	Skill 2	Skill 3	Skill 4	Skill 5	Skill 6	Skill 7	Skill 8
0	Primary	NA	Flight Training Start	0	None	N/A	0	0	0	0	0	0	0	0	0
1	Primary	C21	Contact Cockpit Procedures	4	T6 UTD	SIM	3.9	4	4	3.857142857	4.428571429	0.57143	0.28571	0.28571	0.14285714
2	Primary	C22	Contact EP Trainer	5	T6 OFT	SIM	2.6	3.1428571	3.428571429	3.857142857	4.857142857	0.85714	0.28571	0.28571	0
3	Primary	C31	Contact	4	T6 OFT	SIM	2.6	4	4.142857143	4	4.142857143	1.14286	0.85714	0.57143	0
4	Primary	C41	Contact	10	T-6B	FLT	6	4.7142857	4.428571429	4.142857143	4.285714286	1.14286	1.28571	1	0.85714286
5	Primary	I21	Basic Instruments	1	T6 UTD	SIM	3.9	4.4285714	4	4.285714286	3.285714286	0.42857	0.85714	0.28571	0
6	Primary	C32	Contact	10	T6 OFT	SIM	3.9	4.4285714	4	4.142857143	4.142857143	0.42857	0.71429	0.57143	0
7	Primary	C42	Contact	10	T-6B	FLT	6.4	4.7142857	4.428571429	4.428571429	4.142857143	0.71429	1	0.71429	0.42857143
8	Primary	C43	Day Contact	10	T-6B	FLT	6.8	4.8571429	4.714285714	4.428571429	4.714285714	0.57143	1.14286	0.71429	0.42857143
9	Primary	C44	Midphase Contact Check	10	T-6B	FLT	1.7	4.8571429	4.857142857	4.857142857	5	0.71429	1	0.42857	0.42857143
10	Primary	C45	Contact Solo Flt	10	T-6B	FLT	1.5	5.1428571	5.142857143	4.571428571	4.142857143	0.42857	0.42857	0.14286	0.28571429
11	Primary	I22	Basic Instruments	1	T6 UTD	SIM	3.9	4	4.142857143	4.142857143	3	0.57143	0.14286	0.28571	0.28571429
12	Primary	I31	Radio Instruments	2	T6 OFT	SIM	7.8	4.7142857	4.428571429	4.285714286	2.857142857	0.57143	0.14286	0.14286	0
13	Primary	I41	Radio Instruments	2	T-6B	FLT	6.4	4.4285714	4.428571429	4.285714286	2.285714286	0.42857	0	0	0
14	Primary	I32	Radio Instruments	2	T6 OFT	SIM	6.5	4.7142857	4.428571429	4.285714286	3.285714286	0.42857	0	0	0.28571429
15	Primary	I33	Instrument Navigation	2	T6 OFT	SIM	2.6	4.5714286	4.571428571	4.428571429	2.714285714	0.57143	0	0	0.28571429
16	Primary	I42	Radio Instruments	2	T-6B	FLT	6.4	4.8571429	4.857142857	4.714285714	2.571428571	0.71429	0	0	0
17	Primary	I43	Instrument Navigation	2	T-6B	FLT	8.5	4.8571429	5	5	3	0.71429	0.28571	0	0.28571429
18	Primary	I44	Instrument Check Flight	8	T-6B	FLT	1.7	5	5.571428571	5.142857143	2.857142857	0.42857	0	0	0.28571429
19	Primary	C33	Contact	10	T6 OFT	SIM	2.6	4.5714286	4	4.285714286	4	2.28571	2.42857	1.28571	0
20	Primary	C46	Day Contact	10	T-6B	FLT	6.8	5.1428571	4.285714286	4.428571429	4.285714286	3	2.71429	1.14286	0.28571429
21	Primary	C47	Final Contact Check Flight	10	T-6B	FLT	1.7	5.5714286	4.571428571	4.857142857	4.571428571	2.28571	2.71429	1.14286	0.28571429
22	Primary	C48	Final Contact Solo	10	T-6B	FLT	1.5	5.2857143	4.285714286	4.571428571	2.857142857	0.42857	1.85714	0.57143	0.28571429
23	Primary	C49	UNKNOWN FLIGHT? NOT IN SYL!	10	T-6B	FLT	1.5	4.8571429	4.285714286	4.428571429	2.857142857	0.85714	0.42857	0.28571	0.28571429
24	Primary	N31	Day Navigation	3	T6 OFT	SIM	1.3	4.4285714	4.714285714	4.714285714	2.714285714	0.42857	0.14286	1.57143	0.28571429
25	Primary	N32	Night Navigation	3	T6 OFT	SIM	1.3	4.1428571	4.571428571	4.571428571	2.857142857	0.71429	0	2	0
26	Primary	N41	Day Navigation	3	T-6B	FLT	1.7	4.4285714	4.571428571	4.571428571	2.857142857	0.42857	0.14286	2	0.28571429
27	Primary	N42	Night Navigation	3	T-6B	FLT	1.7	4.5714286	4.428571429	4.714285714	2.714285714	0.42857	0	1.57143	0.28571429
28	Primary	L31	Low Level	7	T6 OFT	SIM	1.3	4.5714286	4.285714286	4.714285714	3	0.42857	0	3.42857	0
29	Primary	L41	Low Level	7	T-6B	FLT	3.2	4.8571429	4.142857143	4.714285714	3.142857143	0.42857	0	3.57143	0.28571429
30	Primary	F31	Formation	12	T6 OFT	SIM	1.3	3.8571429	4.142857143	3.714285714	2.285714286	0.71429	1.71429	0.85714	0

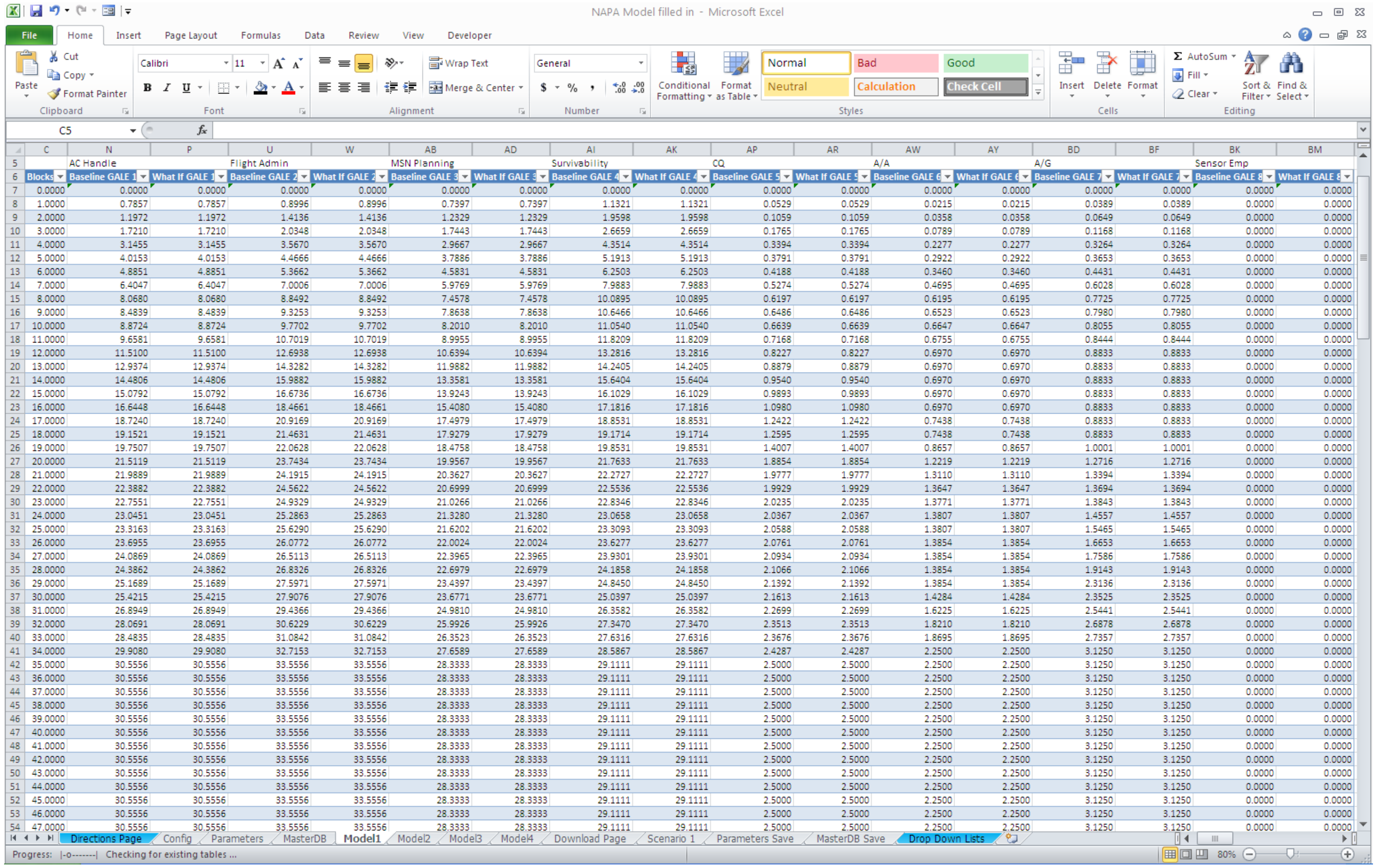
Directions Page | Config | Parameters | MasterDB | Drop Down Lists

Figure 29. First 30 Blocks of Strike Master Database

Once the MasterDB page is filled, the phase models and Download Page are created by calling a subroutine entitled "MakeRemainingPages()" assigned to a button on the MasterDB page shown in Figure 29.

4. Phase Models and Download Page

An individual **Model page** is created for every user-defined phase. The models are generically named to maintain the versatility of the model. An example of a phase model is shown in Figure 30.



	C	N	P	U	W	AB	AD	AI	AK	AP	AR	AW	AY	BD	BF	BK	BM
5		AC Handle		Flight Admin		MSN Planning		Survivability		CQ		A/A		A/G		Sensor Emp	
6	Blocks	Baseline GALE 1	What If GALE 1	Baseline GALE 2	What If GALE 2	Baseline GALE 3	What If GALE 3	Baseline GALE 4	What If GALE 4	Baseline GALE 5	What If GALE 5	Baseline GALE 6	What If GALE 6	Baseline GALE 7	What If GALE 7	Baseline GALE 8	What If GALE 8
7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8	1.0000	0.7857	0.7857	0.8996	0.8996	0.7397	0.7397	1.1321	1.1321	0.0529	0.0529	0.0215	0.0215	0.0389	0.0389	0.0000	0.0000
9	2.0000	1.1972	1.1972	1.4136	1.4136	1.2329	1.2329	1.9598	1.9598	0.1059	0.1059	0.0358	0.0358	0.0649	0.0649	0.0000	0.0000
10	3.0000	1.7210	1.7210	2.0348	2.0348	1.7443	1.7443	2.6659	2.6659	0.1765	0.1765	0.0789	0.0789	0.1168	0.1168	0.0000	0.0000
11	4.0000	3.1455	3.1455	3.5670	3.5670	2.9667	2.9667	4.3514	4.3514	0.3394	0.3394	0.2277	0.2277	0.3264	0.3264	0.0000	0.0000
12	5.0000	4.0153	4.0153	4.4666	4.4666	3.7886	3.7886	5.1913	5.1913	0.3791	0.3791	0.2922	0.2922	0.3653	0.3653	0.0000	0.0000
13	6.0000	4.8851	4.8851	5.3662	5.3662	4.5831	4.5831	6.2503	6.2503	0.4188	0.4188	0.3460	0.3460	0.4431	0.4431	0.0000	0.0000
14	7.0000	6.4047	6.4047	7.0006	7.0006	5.9769	5.9769	7.9883	7.9883	0.5274	0.5274	0.4695	0.4695	0.6028	0.6028	0.0000	0.0000
15	8.0000	8.0680	8.0680	8.8492	8.8492	7.4578	7.4578	10.0895	10.0895	0.6197	0.6197	0.6195	0.6195	0.7725	0.7725	0.0000	0.0000
16	9.0000	8.4839	8.4839	9.3253	9.3253	7.8638	7.8638	10.6466	10.6466	0.6486	0.6486	0.6523	0.6523	0.7980	0.7980	0.0000	0.0000
17	10.0000	8.8724	8.8724	9.7702	9.7702	8.2010	8.2010	11.0540	11.0540	0.6639	0.6639	0.6647	0.6647	0.8055	0.8055	0.0000	0.0000
18	11.0000	9.6581	9.6581	10.7019	10.7019	8.9955	8.9955	11.8209	11.8209	0.7168	0.7168	0.6755	0.6755	0.8444	0.8444	0.0000	0.0000
19	12.0000	11.5100	11.5100	12.6938	12.6938	10.6394	10.6394	13.2816	13.2816	0.8227	0.8227	0.6970	0.6970	0.8833	0.8833	0.0000	0.0000
20	13.0000	12.9374	12.9374	14.3282	14.3282	11.9882	11.9882	14.2405	14.2405	0.8879	0.8879	0.6970	0.6970	0.8833	0.8833	0.0000	0.0000
21	14.0000	14.4806	14.4806	15.9882	15.9882	13.3581	13.3581	15.6404	15.6404	0.9540	0.9540	0.6970	0.6970	0.8833	0.8833	0.0000	0.0000
22	15.0000	15.0792	15.0792	16.6736	16.6736	13.9243	13.9243	16.1029	16.1029	0.9893	0.9893	0.6970	0.6970	0.8833	0.8833	0.0000	0.0000
23	16.0000	16.6448	16.6448	18.4661	18.4661	15.4080	15.4080	17.1816	17.1816	1.0980	1.0980	0.6970	0.6970	0.8833	0.8833	0.0000	0.0000
24	17.0000	18.7240	18.7240	20.9169	20.9169	17.4979	17.4979	18.8531	18.8531	1.2422	1.2422	0.7438	0.7438	0.8833	0.8833	0.0000	0.0000
25	18.0000	19.1521	19.1521	21.4631	21.4631	17.9279	17.9279	19.1714	19.1714	1.2595	1.2595	0.7438	0.7438	0.8833	0.8833	0.0000	0.0000
26	19.0000	19.7507	19.7507	22.0628	22.0628	18.4758	18.4758	19.8531	19.8531	1.4007	1.4007	0.8657	0.8657	1.0001	1.0001	0.0000	0.0000
27	20.0000	21.5119	21.5119	23.7434	23.7434	19.9567	19.9567	21.7633	21.7633	1.8854	1.8854	1.2219	1.2219	1.2716	1.2716	0.0000	0.0000
28	21.0000	21.9889	21.9889	24.1915	24.1915	20.3627	20.3627	22.2727	22.2727	1.9777	1.9777	1.3110	1.3110	1.3394	1.3394	0.0000	0.0000
29	22.0000	22.3882	22.3882	24.5622	24.5622	20.6999	20.6999	22.5536	22.5536	1.9929	1.9929	1.3647	1.3647	1.3694	1.3694	0.0000	0.0000
30	23.0000	22.7551	22.7551	24.9329	24.9329	21.0266	21.0266	22.8346	22.8346	2.0235	2.0235	1.3771	1.3771	1.3843	1.3843	0.0000	0.0000
31	24.0000	23.0451	23.0451	25.2863	25.2863	21.3280	21.3280	23.0658	23.0658	2.0367	2.0367	1.3807	1.3807	1.4557	1.4557	0.0000	0.0000
32	25.0000	23.3163	23.3163	25.6290	25.6290	21.6202	21.6202	23.3093	23.3093	2.0588	2.0588	1.3807	1.3807	1.5465	1.5465	0.0000	0.0000
33	26.0000	23.6955	23.6955	26.0772	26.0772	22.0024	22.0024	23.6277	23.6277	2.0761	2.0761	1.3854	1.3854	1.6653	1.6653	0.0000	0.0000
34	27.0000	24.0869	24.0869	26.5113	26.5113	22.3965	22.3965	23.9301	23.9301	2.0934	2.0934	1.3854	1.3854	1.7586	1.7586	0.0000	0.0000
35	28.0000	24.3862	24.3862	26.8326	26.8326	22.6979	22.6979	24.1858	24.1858	2.1066	2.1066	1.3854	1.3854	1.9143	1.9143	0.0000	0.0000
36	29.0000	25.1689	25.1689	27.5971	27.5971	23.4397	23.4397	24.8450	24.8450	2.1392	2.1392	1.3854	1.3854	2.3136	2.3136	0.0000	0.0000
37	30.0000	25.4215	25.4215	27.9076	27.9076	23.6771	23.6771	25.0397	25.0397	2.1613	2.1613	1.4284	1.4284	2.3525	2.3525	0.0000	0.0000
38	31.0000	26.8949	26.8949	29.4366	29.4366	24.9810	24.9810	26.3582	26.3582	2.2699	2.2699	1.6225	1.6225	2.5441	2.5441	0.0000	0.0000
39	32.0000	28.0691	28.0691	30.6229	30.6229	25.9926	25.9926	27.3470	27.3470	2.3513	2.3513	1.8210	1.8210	2.6878	2.6878	0.0000	0.0000
40	33.0000	28.4835	28.4835	31.0842	31.0842	26.3523	26.3523	27.6316	27.6316	2.3676	2.3676	1.8695	1.8695	2.7357	2.7357	0.0000	0.0000
41	34.0000	29.9080	29.9080	32.7153	32.7153	27.6589	27.6589	28.5867	28.5867	2.4287	2.4287	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
42	35.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
43	36.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
44	37.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
45	38.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
46	39.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
47	40.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
48	41.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
49	42.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
50	43.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
51	44.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
52	45.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
53	46.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000
54	47.0000	30.5556	30.5556	33.5556	33.5556	28.3333	28.3333	29.1111	29.1111	2.5000	2.5000	2.2500	2.2500	3.1250	3.1250	0.0000	0.0000

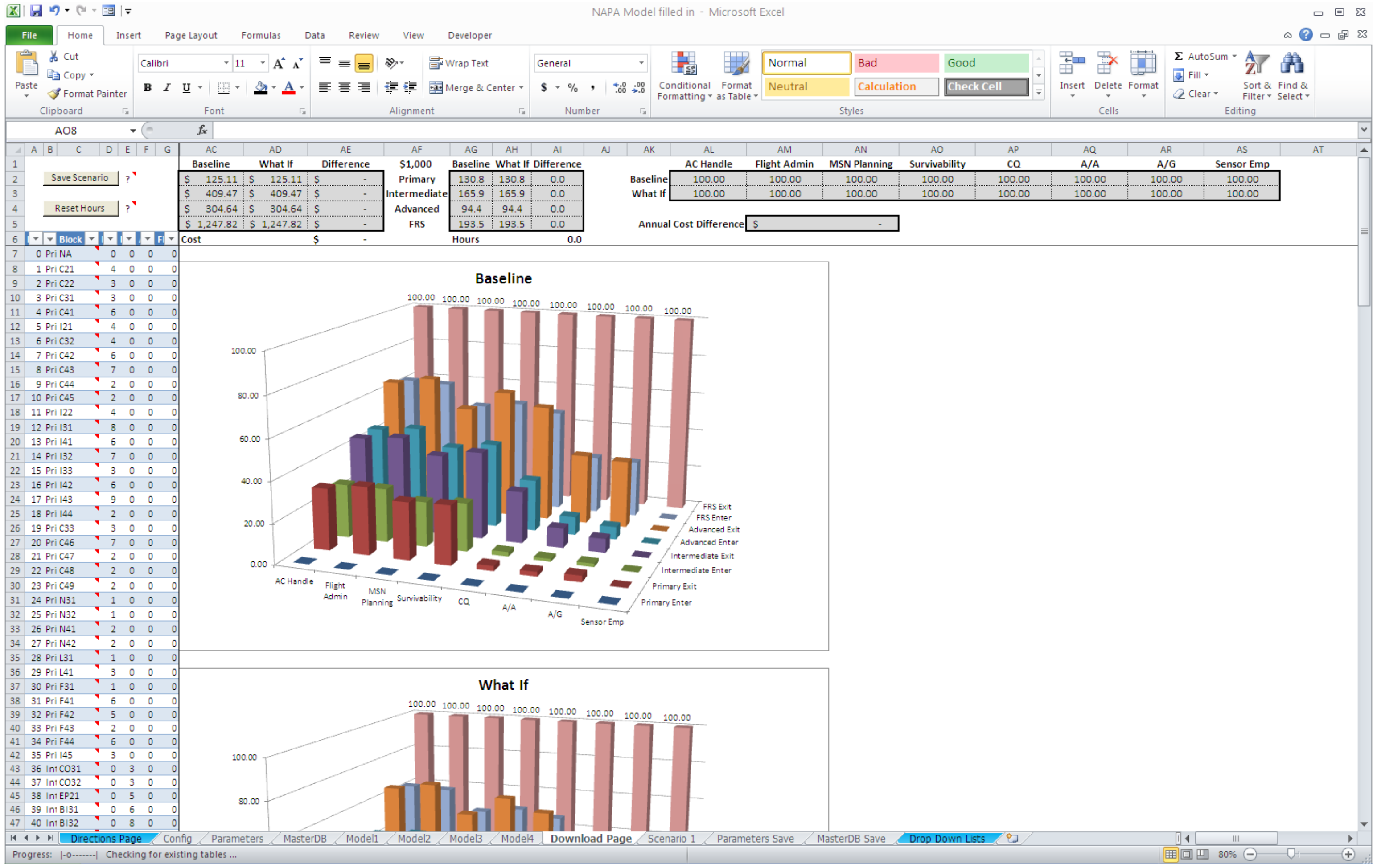
Figure 30. Initial Phase Model Interface Example

Many columns containing background information and data manipulation are automatically hidden during the subroutine to make the output more user-friendly. Upon completion, only the block number and columns associated with baseline and what-if values are displayed. This allows the user to quickly create a graphical depiction of the data, should one be desired. The ability to quickly graph these outputs also provides a quick way to re-create the original GALE learning curves.

The **Download page** is where the majority of user interaction takes place once the final pages are created. There are three main sections on the Download page:

- *Scenario input section.* Located on the far left of the Download page, this section is where users input their what-if scenarios. This section takes advantage of inherent Excel table properties and allows filtering to focus on concentration areas. Additionally, conditional formatting in the what-if columns will highlight when a change or difference from the baseline has been entered.
- *Real-time analysis section.* Located on the top of the Download page, this section provides an instantaneous cost and hour difference within the phases along with the difference in skill proficiency compared to the baseline.
- *Chart section.* This section shows the 3-D charts for baseline and what-if scenarios.

The initial view of the Download page is shown in Figure 31.



Baseline	What If	Difference	$1,000	Baseline	What If	Difference
$ 125.11	$ 125.11	$ -	Primary	130.8	130.8	0.0
$ 409.47	$ 409.47	$ -	Intermediate	165.9	165.9	0.0
$ 304.64	$ 304.64	$ -	Advanced	94.4	94.4	0.0
$ 1,247.82	$ 1,247.82	$ -	FRS	193.5	193.5	0.0
Cost		$ -		Hours		0.0

	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Baseline	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
What If	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Annual Cost Difference $ -

	Block				
0	Pri NA	0	0	0	0
1	Pri C21	4	0	0	0
2	Pri C22	3	0	0	0
3	Pri C31	3	0	0	0
4	Pri C41	6	0	0	0
5	Pri I21	4	0	0	0
6	Pri C32	4	0	0	0
7	Pri C42	6	0	0	0
8	Pri C43	7	0	0	0
9	Pri C44	2	0	0	0
10	Pri C45	2	0	0	0
11	Pri I22	4	0	0	0
12	Pri I31	8	0	0	0
13	Pri I41	6	0	0	0
14	Pri I32	7	0	0	0
15	Pri I33	3	0	0	0
16	Pri I42	6	0	0	0
17	Pri I43	9	0	0	0
18	Pri I44	2	0	0	0
19	Pri C33	3	0	0	0
20	Pri C46	7	0	0	0
21	Pri C47	2	0	0	0
22	Pri C48	2	0	0	0
23	Pri C49	2	0	0	0
24	Pri N31	1	0	0	0
25	Pri N32	1	0	0	0
26	Pri N41	2	0	0	0
27	Pri N42	2	0	0	0
28	Pri L31	1	0	0	0
29	Pri L41	3	0	0	0
30	Pri F31	1	0	0	0
31	Pri F41	6	0	0	0
32	Pri F42	5	0	0	0
33	Pri F43	2	0	0	0
34	Pri F44	6	0	0	0
35	Pri I45	3	0	0	0
36	Int CO31	0	3	0	0
37	Int CO32	0	3	0	0
38	Int EP21	0	5	0	0
39	Int BI31	0	6	0	0
40	Int BI32	0	8	0	0

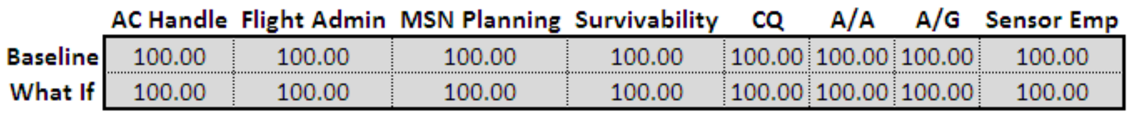
Progress: |-o-------| Checking for existing tables ...

Figure 31. Initial Download Page Interface

B. MODEL OUTPUTS

1. Training Pipeline Baseline Output (Strike Example)

The NAPA model provides Baseline skill proficiencies in both table and graphical formats (Figures 32 and 33 respectively).



	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Baseline	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
What If	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Figure 32. NAPA Model Baseline Proficiency Table

Both Baseline and "What If" scenarios initially show 100% proficiency gain, indicating that no exploratory values have been input into the download page.

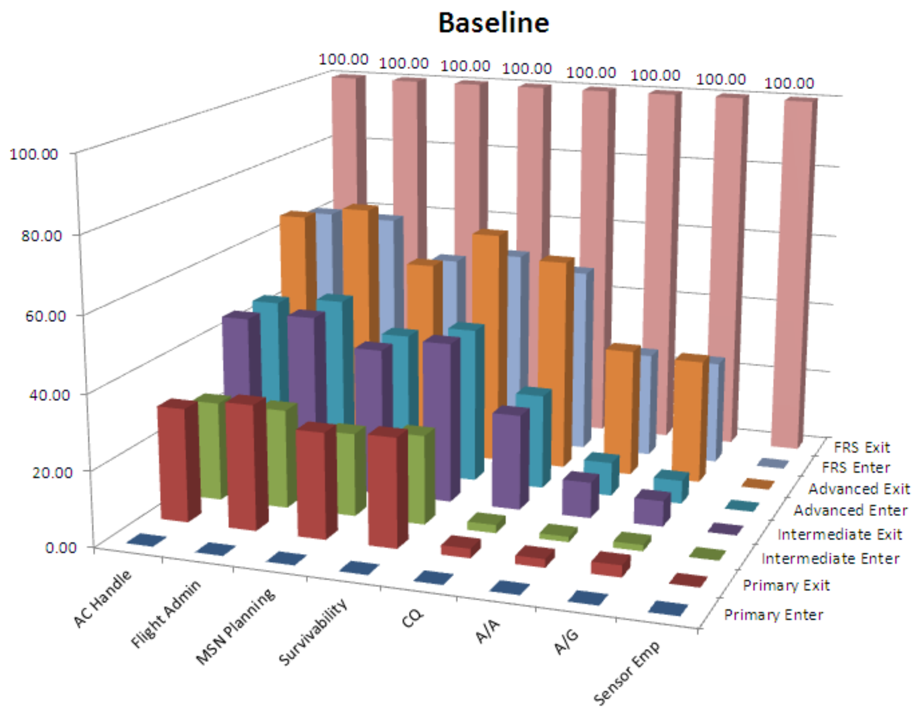


Figure 33. NAPA Model Output of Baseline Skill Proficiencies

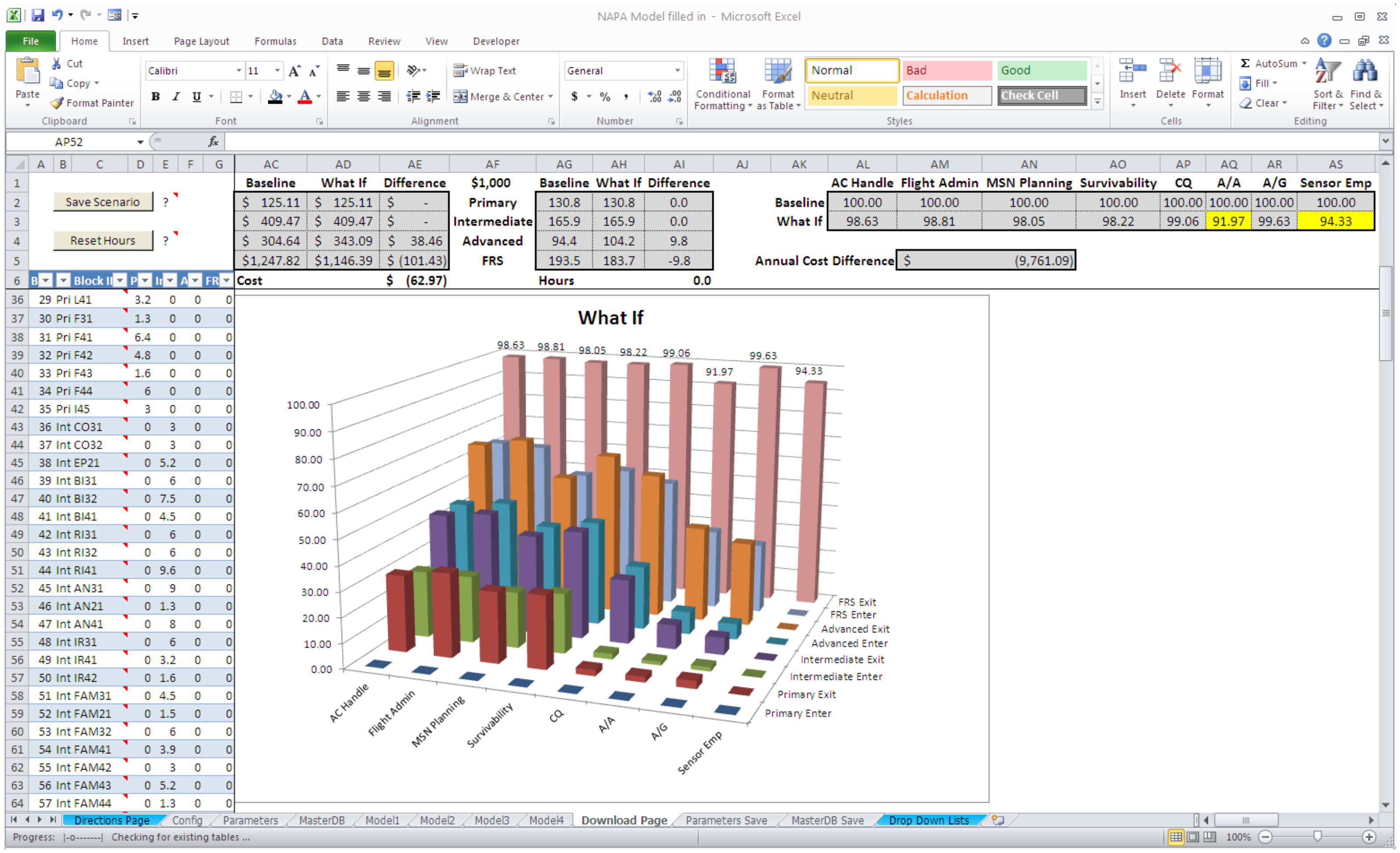
The graphical output plots the proficiency levels (vertical axis) of identified pipeline skills (horizontal axis) across chronological phase entry and exit points throughout the pipeline (into page). This graphic enforces a couple critical components of the Strike training pipeline:

- Similar to the manual example of the proficiency curves provided by NAWCTSD HP/ISD, skill proficiency is gained at different rates throughout the training pipeline. Subsequently, reductions or alterations in the syllabus at specific times may have a greater effect on one skill's proficiency gain than on another's.
- The baseline assumption is that the current pipeline provides 100% of the proficiency required for a pilot to become a productive member of the Fleet forces.

The overall shape of the surface created by the varied proficiency gains depict where potential capability gaps exist. If proficiency gain in a skill is only provided in one phase, any change in that phase potentially creates an unrecoverable proficiency deficit in that skill.

2. Downloading Example

To provide an example of how the NAPA model reacts to a downloading scenario, almost ten hours from the FRS phase are selected to download to the Advanced phase. Specifically, Basic Fighter Maneuvers (BFM) blocks 162–166 are removed from the FRS phase and added to the Advanced phase. Figure 34 shows the Download page example.



Baseline	What If	Difference	$1,000	Baseline	What If	Difference
$ 125.11	$ 125.11	$ -	Primary	130.8	130.8	0.0
$ 409.47	$ 409.47	$ -	Intermediate	165.9	165.9	0.0
$ 304.64	$ 343.09	$ 38.46	Advanced	94.4	104.2	9.8
$1,247.82	$1,146.39	$ (101.43)	FRS	193.5	183.7	-9.8
Cost		$ (62.97)		Hours		0.0

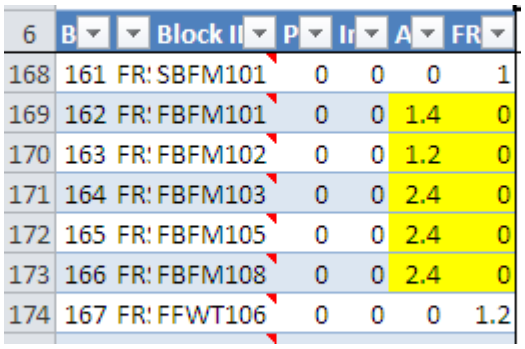
	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Baseline	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
What If	98.63	98.81	98.05	98.22	99.06	91.97	99.63	94.33

Annual Cost Difference $ (9,761.09)

B		Block II	P	Ir	A	FR
29	Pri	L41	3.2	0	0	0
30	Pri	F31	1.3	0	0	0
31	Pri	F41	6.4	0	0	0
32	Pri	F42	4.8	0	0	0
33	Pri	F43	1.6	0	0	0
34	Pri	F44	6	0	0	0
35	Pri	I45	3	0	0	0
36	Int	CO31	0	3	0	0
37	Int	CO32	0	3	0	0
38	Int	EP21	0	5.2	0	0
39	Int	BI31	0	6	0	0
40	Int	BI32	0	7.5	0	0
41	Int	BI41	0	4.5	0	0
42	Int	RI31	0	6	0	0
43	Int	RI32	0	6	0	0
44	Int	RI41	0	9.6	0	0
45	Int	AN31	0	9	0	0
46	Int	AN21	0	1.3	0	0
47	Int	AN41	0	8	0	0
48	Int	IR31	0	6	0	0
49	Int	IR41	0	3.2	0	0
50	Int	IR42	0	1.6	0	0
51	Int	FAM31	0	4.5	0	0
52	Int	FAM21	0	1.5	0	0
53	Int	FAM32	0	6	0	0
54	Int	FAM41	0	3.9	0	0
55	Int	FAM42	0	3	0	0
56	Int	FAM43	0	5.2	0	0
57	Int	FAM44	0	1.3	0	0

Figure 34. Partial View of Download Page Contents During Scenario Analysis

A more detailed look into parts of the NAPA model reveals how it reacts to the scenario. Figure 35 focuses on the blocks that were altered and shows how the yellow highlights used by the model to depict an hour-for-hour block download in the scenario input section.



6	B	Block I	P	Ir	A	FR
168	161	FR: SBFM101	0	0	0	1
169	162	FR: FBFM101	0	0	1.4	0
170	163	FR: FBFM102	0	0	1.2	0
171	164	FR: FBFM103	0	0	2.4	0
172	165	FR: FBFM105	0	0	2.4	0
173	166	FR: FBFM108	0	0	2.4	0
174	167	FR: FFWT106	0	0	0	1.2

Figure 35. Portion of Download Page Depicting Downloaded BFM Blocks from the FRS to Advanced

The What if chart (Figure 36) in the chart section of the Download page shows the skill proficiency levels resulting from downloading the BFM blocks.

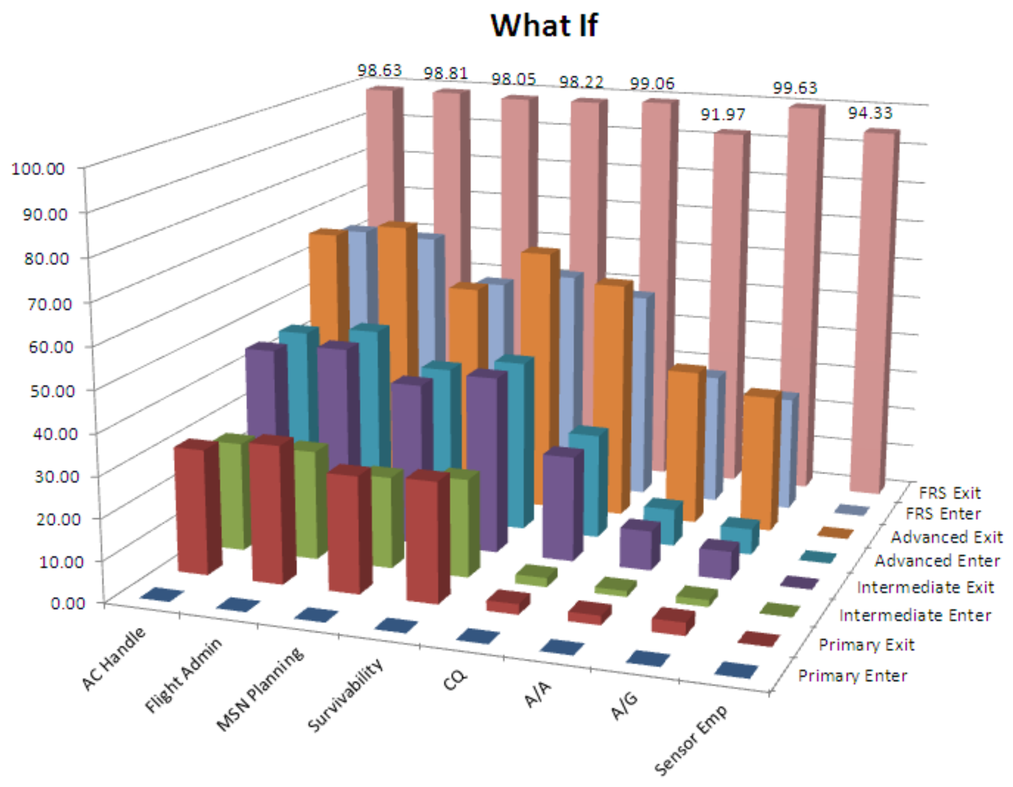
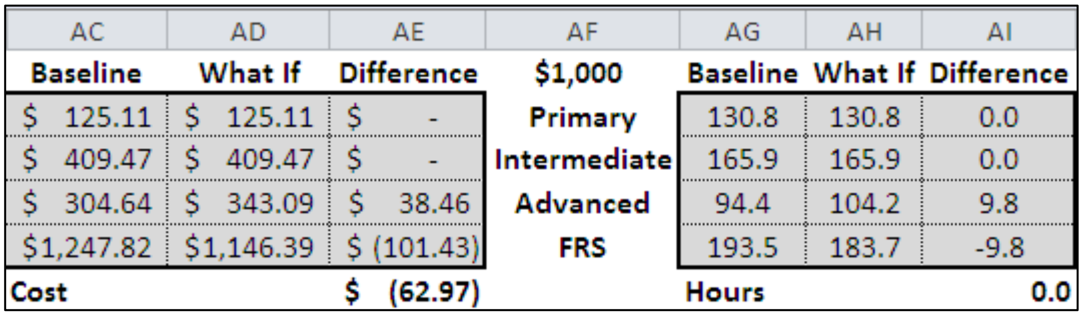


Figure 36. Download Page What If Chart: Skill Proficiency Levels Resulting From Downloaded Blocks

Like the baseline chart, the graphical output plots the proficiency levels (vertical axis) of identified pipeline skills (horizontal axis) across chronological phase entry and exit points throughout the pipeline (into page). The variation in output skill proficiencies resulting from the scenario shows the individual dependency each skill has with syllabus hours.

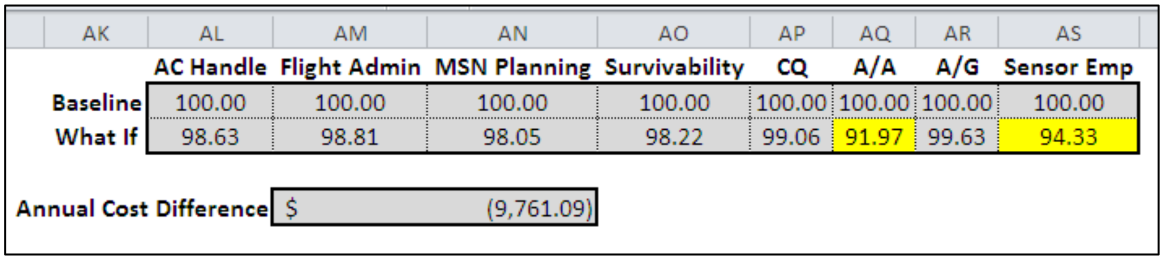
In the real-time analysis section, both cost and hour difference resulting from the download is calculated. In this example, by moving 9.8 hours from the FRS phase to the Advanced phase, results in a savings of $62,970 per pilot. The total hours in the pipeline remain the same, but the downloaded hours are shifted from FRS to Advanced phases. The hour shifts as well as the cost savings can be seen in Figure 37 (part of the Download page).



AC	AD	AE	AF	AG	AH	AI
Baseline	What If	Difference	$1,000	Baseline	What If	Difference
$ 125.11	$ 125.11	$ -	Primary	130.8	130.8	0.0
$ 409.47	$ 409.47	$ -	Intermediate	165.9	165.9	0.0
$ 304.64	$ 343.09	$ 38.46	Advanced	94.4	104.2	9.8
$1,247.82	$1,146.39	$ (101.43)	FRS	193.5	183.7	-9.8
Cost		$ (62.97)		Hours		0.0

Figure 37. Download Page Cost Table and Hours Table Resulting From Downloaded Blocks

In Figure 38, proficiency threshold breeches are highlighted to indicate when the resulting skill proficiency drops below the user-defined threshold. Both air-to-air (A/A) and Sensor Employment (Sensor Emp) blocks show breeches in this scenario—the proficiency threshold was set for 95% in this example.



AK	AL	AM	AN	AO	AP	AQ	AR	AS
	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Baseline	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
What If	98.63	98.81	98.05	98.22	99.06	91.97	99.63	94.33

Annual Cost Difference	$ (9,761.09)

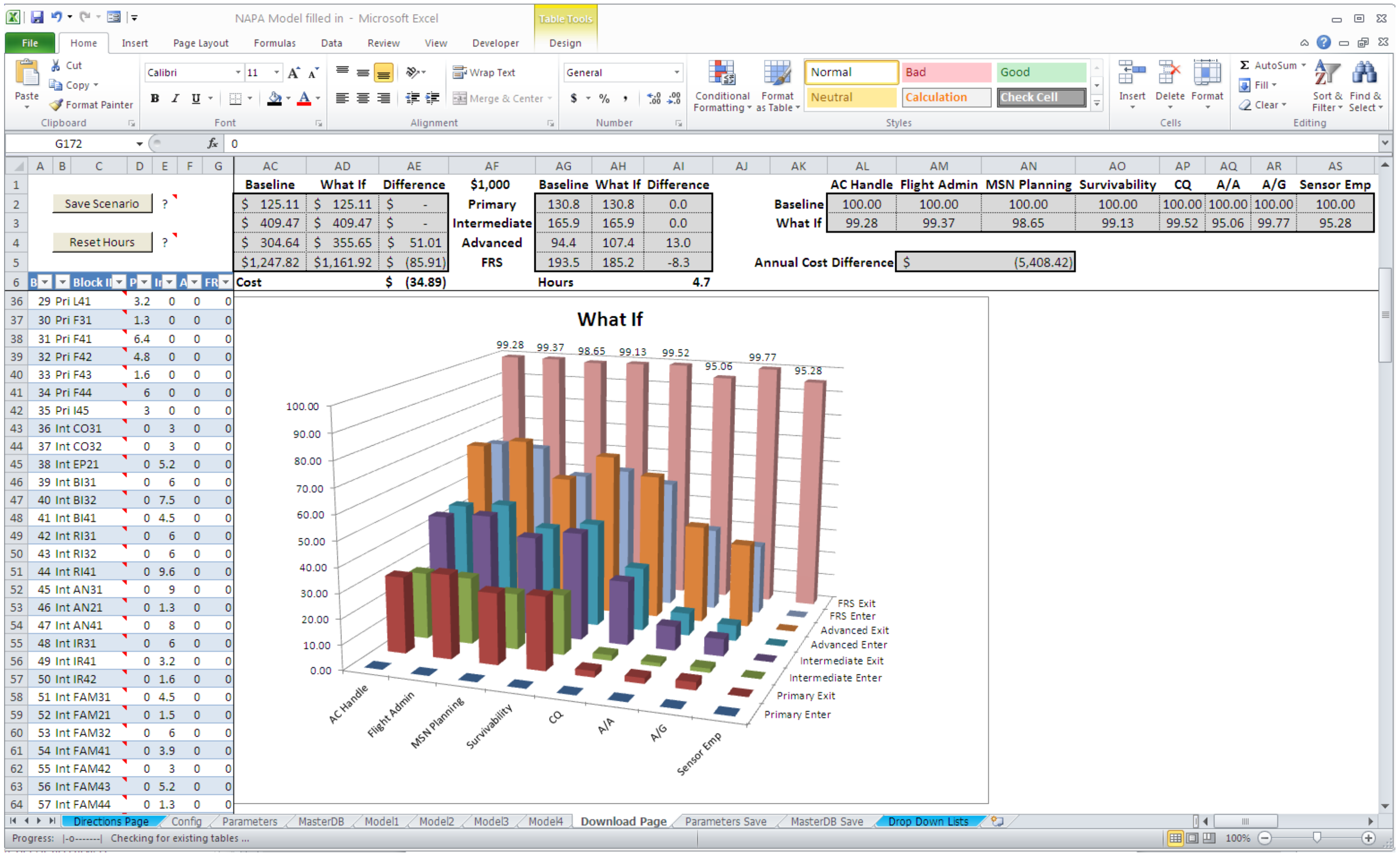
Figure 38. Download Page Proficiency Table and Annual Cost Difference Resulting From Downloaded Blocks

Additionally, the annual cost difference is provided to the user based on CONFIG page throughput values and cost differences associated with the specific scenario.

The one-for-one download scenario—where the same amount of block hours are moved into a different phase—will almost always produce some level of proficiency degradation. In some cases, proficiency in skills may be reduced to unacceptable levels and may be un-recoverable without adding hours to maintain skill proficiency at minimum threshold levels.

3. Maintaining Threshold Levels

This example shows how users might utilize the model to investigate a syllabus solution to maintain their minimum proficiency values. Using the previous download scenario as a starting point, this scenario will add hours where necessary to maintain the desired threshold levels. Figure 39 depicts what the user would see on the Download page during this type of scenario analysis.



	AC	AD	AE	AF	AG	AH	AI
1	Baseline	What If	Difference	$1,000	Baseline	What If	Difference
2	$ 125.11	$ 125.11	$ -	Primary	130.8	130.8	0.0
3	$ 409.47	$ 409.47	$ -	Intermediate	165.9	165.9	0.0
4	$ 304.64	$ 355.65	$ 51.01	Advanced	94.4	107.4	13.0
5	$1,247.82	$1,161.92	$ (85.91)	FRS	193.5	185.2	-8.3
6	Cost		$ (34.89)		Hours		4.7

	AL	AM	AN	AO	AP	AQ	AR	AS
	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Baseline	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
What If	99.28	99.37	98.65	99.13	99.52	95.06	99.77	95.28

Annual Cost Difference $ (5,408.42)

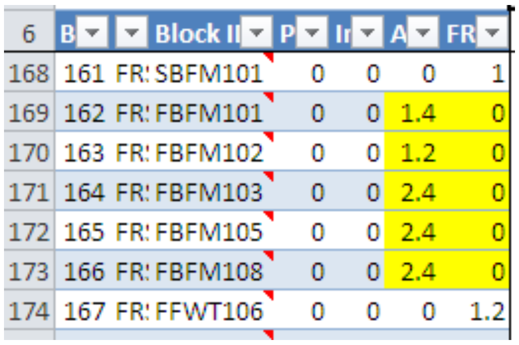
	B	Block II	P	Ir	A	FR
36	29	Pri L41	3.2	0	0	0
37	30	Pri F31	1.3	0	0	0
38	31	Pri F41	6.4	0	0	0
39	32	Pri F42	4.8	0	0	0
40	33	Pri F43	1.6	0	0	0
41	34	Pri F44	6	0	0	0
42	35	Pri I45	3	0	0	0
43	36	Int CO31	0	3	0	0
44	37	Int CO32	0	3	0	0
45	38	Int EP21	0	5.2	0	0
46	39	Int BI31	0	6	0	0
47	40	Int BI32	0	7.5	0	0
48	41	Int BI41	0	4.5	0	0
49	42	Int RI31	0	6	0	0
50	43	Int RI32	0	6	0	0
51	44	Int RI41	0	9.6	0	0
52	45	Int AN31	0	9	0	0
53	46	Int AN21	0	1.3	0	0
54	47	Int AN41	0	8	0	0
55	48	Int IR31	0	6	0	0
56	49	Int IR41	0	3.2	0	0
57	50	Int IR42	0	1.6	0	0
58	51	Int FAM31	0	4.5	0	0
59	52	Int FAM21	0	1.5	0	0
60	53	Int FAM32	0	6	0	0
61	54	Int FAM41	0	3.9	0	0
62	55	Int FAM42	0	3	0	0
63	56	Int FAM43	0	5.2	0	0
64	57	Int FAM44	0	1.3	0	0

Progress: |-o-------| Checking for existing tables ...

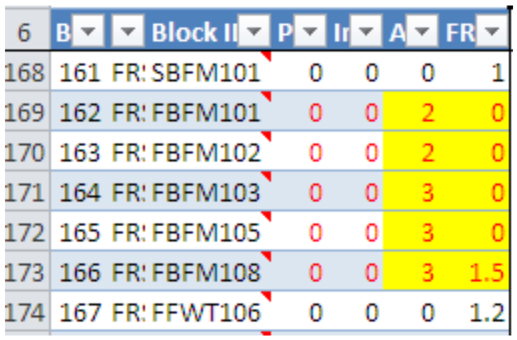
Figure 39. Partial View of Download Page Contents

No highlighted cells in the real-time analysis section indicate that the thresholds for skill proficiencies are met.

A closer look into the scenario input section (Figure 40) shows that additional hours have been added to the Advanced syllabus to compensate for the loss in proficiency in the Air-to-Air skill. 1.5 hours have been re-inserted into the FRS syllabus to increase the Sensor Employment skill proficiency above threshold levels. Numbers in the model are red, indicating a change in the number of hours in that block.



6	B		Block II	P	Ir	A	FR
168	161	FR:	SBFM101	0	0	0	1
169	162	FR:	FBFM101	0	0	1.4	0
170	163	FR:	FBFM102	0	0	1.2	0
171	164	FR:	FBFM103	0	0	2.4	0
172	165	FR:	FBFM105	0	0	2.4	0
173	166	FR:	FBFM108	0	0	2.4	0
174	167	FR:	FFWT106	0	0	0	1.2



6	B		Block II	P	Ir	A	FR
168	161	FR:	SBFM101	0	0	0	1
169	162	FR:	FBFM101	0	0	2	0
170	163	FR:	FBFM102	0	0	2	0
171	164	FR:	FBFM103	0	0	3	0
172	165	FR:	FBFM105	0	0	3	0
173	166	FR:	FBFM108	0	0	3	1.5
174	167	FR:	FFWT106	0	0	0	1.2

Figure 40. Original Download Scenario (Top) and Altered Scenario to Maintain Minimum Proficiency Values (Bottom)

The following alterations in hours are made in this example:

- Block 162 hours increased from 1.4 hours to 2 hours in advanced
- Block 163 hours increased from 1.2 hours to 2 hours in advanced
- Block 164–166 hours increased from 2.4 hours to 3 hours in advanced
- Block 166 hours added from 0 to 1.5 hours in FRS

The altered scenario output is also graphically available in the chart section (Figure 41).

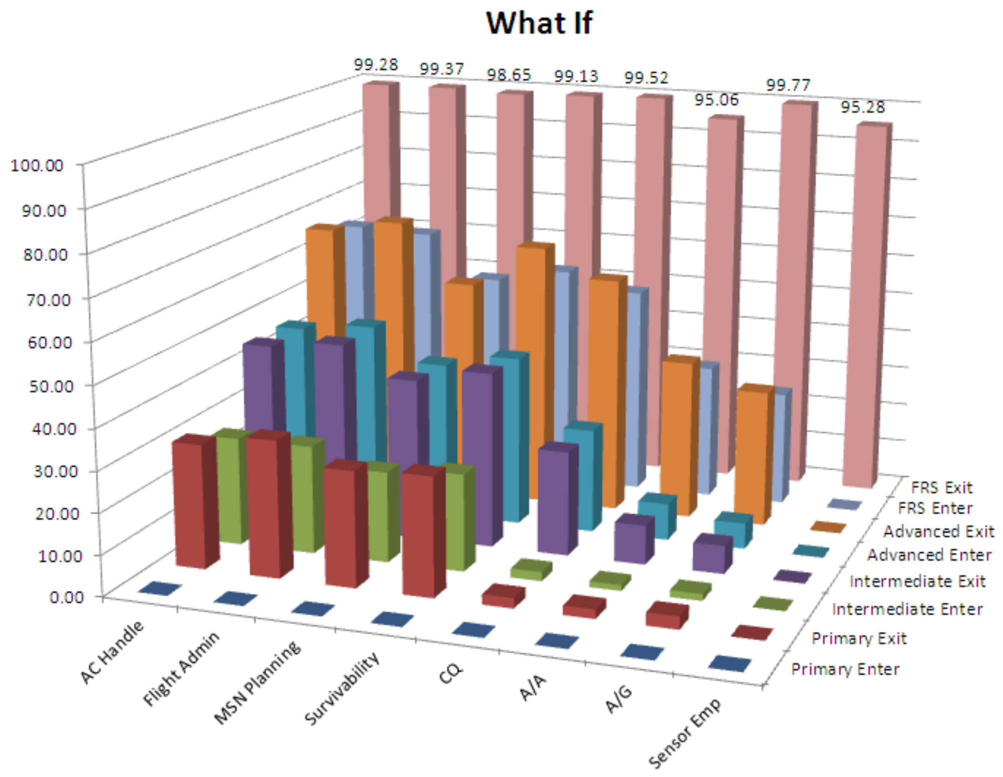
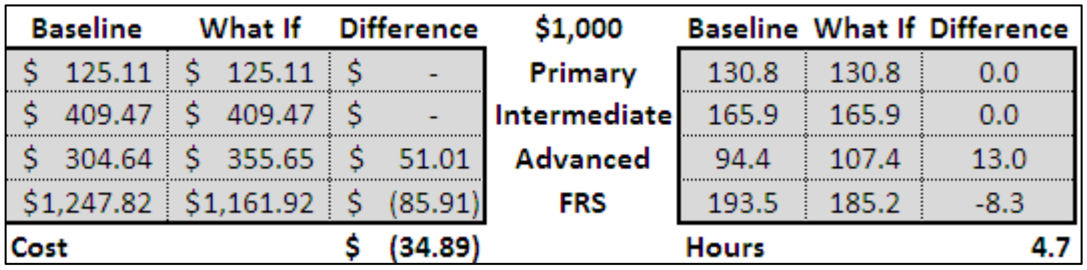


Figure 41. Download Page What If Chart Output

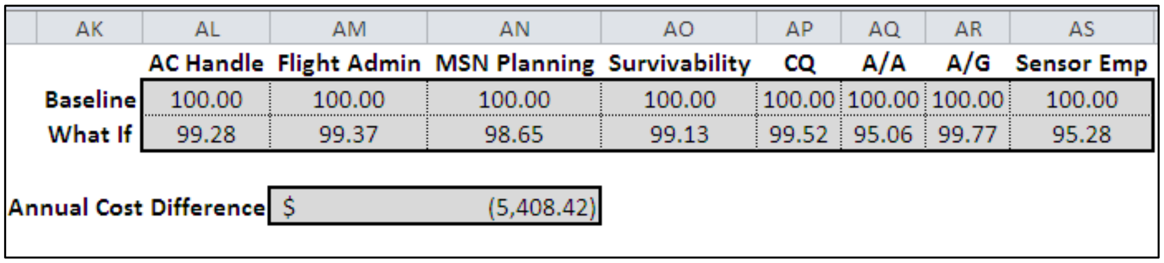
The real-time analysis section shows the change in hours for the two affected phases. Figure 42 shows that 13 hours have been added to the advanced phase, but 8.3 hours have been removed from the FRS phase, saving $34,890 per pilot.



Baseline	What If	Difference	$1,000	Baseline	What If	Difference
$ 125.11	$ 125.11	$ -	Primary	130.8	130.8	0.0
$ 409.47	$ 409.47	$ -	Intermediate	165.9	165.9	0.0
$ 304.64	$ 355.65	$ 51.01	Advanced	94.4	107.4	13.0
$1,247.82	$1,161.92	$ (85.91)	FRS	193.5	185.2	-8.3
Cost		$ (34.89)		Hours		4.7

Figure 42. Download Page Cost Table and Hour Table for Threshold Example

A close-up look at the rest of the real-time analysis section (Figure 43) shows that no proficiency values breeched the desired threshold (no highlights) and an annual cost savings of $5,408,420 based on a throughput of 155 pilots in the Strike pipeline.



AK	AL	AM	AN	AO	AP	AQ	AR	AS
	AC Handle	Flight Admin	MSN Planning	Survivability	CQ	A/A	A/G	Sensor Emp
Baseline	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
What If	99.28	99.37	98.65	99.13	99.52	95.06	99.77	95.28

Annual Cost Difference	$ (5,408.42)

Figure 43. Download Page Proficiency Table and Annual Cost Difference for Threshold Example

The completed model is capable of supporting NAWCTSD HP/ISD efforts to analyze the effect that alterations in flight hours have on the proficiency of aircrew as well as the cost required to obtain that proficiency.

APPENDIX E: COMPLETE VBA APPLICATION CODE

The VBA code is provided, in its entirety, for historical reference. The code is commented to help determine the functionality each subroutine provides the NAPA model.

BEGIN CODE:

```
Option Explicit

‘ This DEBUGFLAG affects many steps:
‘ (1) in MakeParametersPage, if False then all dependent worksheets are
‘     found/deleted; if True, only “Parameters” is found/deleted.
‘ (2) in many functions, if False then non-user-dependent worksheets are
‘     hidden from view; if True, no worksheets are hidden
‘ (3) in MakeMasterDB and MakeModel, if False then non-user-dependent columns
‘     are hidden; if True, none of the columns are hidden
Public Const DEBUGFLAG As Boolean = False

‘ Starting cells for various pages
Public Const STARTCELL_PARAMETERS As String = “C6”
Public Const STARTCELL_MASTERDB As String = “C6”
Public Const STARTCELL_DLPAGE As String = “A6”
Public Const STARTCELL_MODEL As String = “C6”

Public Const CFG_FUNCLEN As Integer = 40

‘ These refer to the PhaseRef table on the Config page.
Public Const PR_PHASE As Integer = 1
Public Const PR_PLATFORM As Integer = 2
Public Const PR_MAXHRFACTOR As Integer = 3
Public Const PR_COSTFLT As Integer = 4
Public Const PR_COSTSIM As Integer = 5
Public Const PR_BLOCKSPERPHASE As Integer = 6

Public Const PR_BTN_MASTERDB As String = “A2:B3”
Public Const PR_BTN_MASTERDB_HELP As String = “C2”
Public Const MDB_BTN_DLPAGE As String = “A2:C3”
Public Const MDB_BTN_DLPAGE_HELP As String = “D2”

Public Const MDB_BLOCKS As Integer = 1
Public Const MDB_PHASE As Integer = 2
Public Const MDB_BLOCKID As Integer = 3
Public Const MDB_TITLE As Integer = 4
Public Const MDB_FUNCTION As Integer = 5
Public Const MDB_MEDIUM As Integer = 6
Public Const MDB_FLTORSIM As Integer = 7
Public Const MDB_BASELINEHOURS As Integer = 8
Public Const MDB_SKILLS As Integer = 9
Public Const MDB_NUMCOL As Integer = 8 ‘ set to max of other MDB_* (except
MDB_SKILLS which is variable)

‘ These refer to the DownloadTable on the Download Table page
Public Const DL_BLOCKS As Integer = 1
Public Const DL_PHASE As Integer = 2
Public Const DL_BLOCKID As Integer = 3
Public Const DL_PHASES As Integer = 4
‘ these are defined to be AFTER the DL_PHASES set of columns
Public Const DL_BASELINEHRS As Integer = 1
Public Const DL_TITLE As Integer = 2
Public Const DL_COSTDIFF As Integer = 3
Public Const DL_HOURDIFF As Integer = 4
Public Const DL_ORIGPHASES As Integer = 5

Public Const DL_BTN_SAVE As String = “B2:D2”
Public Const DL_BTN_SAVE_HELP As String = “E2”
Public Const DL_BTN_RESET As String = “B4:D4”
Public Const DL_BTN_RESET_HELP As String = “E4”

Public Const DL_CHART_LEFT As Double = 2032
Public Const DL_CHART_TOP As Double = 115
Public Const DL_CHART_WIDTH As Double = 600
Public Const DL_CHART_HEIGHT As Double = 425

Public Const SCENARIO_TIMEDATE As String = “C2”

Public Const FMT_ACCT As String = “_($* #,##0.00_);_($* (#,##0.00);_($* ““-
”“??_);_(@_)”

Public SB_Message As String
Public SB_Index As Integer
Public SB_HeartBeat As Variant


‘ **********************************************************
‘ UpdateConfigPage()
‘ **********************************************************
Sub UpdateConfigPage()
   Dim rPhaseCount As Range, rPhaseRef As Range, rFuncListStart As Range
   Dim iIndexP As Integer, iIndex As Integer
   Dim sPhases() As String, sLineCounts() As String, sPhaseNames() As Variant
   Dim iOldCount As Integer

   Set rPhaseCount = Range("PhaseCount")
   If rPhaseCount.Value < 1 Then
      rPhaseCount.Value = 1
   End If

   Set rPhaseRef = Range("PhaseRef")
   Set rFuncListStart = Range("FuncListStart")

   ' ********************************************************
   ' Phases Definition

   iOldCount = rPhaseRef.End(xlDown).Row - rPhaseRef.Row + 1

   'ReDim sPhaseNames(1 To iOldCount) As Variant
   'sPhaseNames = Application.Transpose(rPhaseRef.Resize(iOldCount, 1).Value)

   ' preserve the phase names in the FuncList table
   rFuncListStart.Offset(-1, 1).Resize(columnsize:=iOldCount).Value = _
      Application.Transpose(rPhaseRef.Resize(iOldCount, 1).Value)

   With rPhaseRef.CurrentRegion
      .ClearFormats
      .Validation.Delete
   End With
   rPhaseRef.Offset(0,3).Resize(WorksheetFunction.Max(rPhaseCount.Value,
iOldCount), 5).ClearContents

   If iOldCount < rPhaseCount.Value Then
      ' insert new-old rows after old
      rPhaseRef.Offset(iOldCount,0).Resize(rowsize:=rPhaseCount.Value -
iOldCount).Insert shift:=xlDown
      For iIndexP = iOldCount + 1 To rPhaseCount.Value
         rPhaseRef(iIndexP, 1).Value = "Phase " & iIndexP
      Next iIndexP
   ElseIf iOldCount > rPhaseCount.Value Then
      ' insert 1 row after new
      rPhaseRef.Offset(rPhaseCount.Value, 0).Resize(rowsize:=1).Insert shift:=xlDown
   End If

   Set rPhaseRef = Range("PhaseRef")
   ‘ formulas for the non-user-editable cells
   For iIndexP = 1 To rPhaseCount.Value
      If rPhaseRef(iIndexP, 1).Value = ““ Then
         rPhaseRef(iIndexP, 1).Value = “Phase “ & iIndexP
      End If
      With rPhaseRef.Offset(iIndexP - 1, 3).Resize(1, 5)
         .FormulaR1C1 = Array( _
            “=IF(ISTEXT(RC[-2]), VLOOKUP(RC[-2], PLATFORMLIST, 2, FALSE),
““““),” _
            “=IF(ISTEXT(RC[-3]), VLOOKUP(RC[-3], PLATFORMLIST, 3, FALSE),
““““),” _
            “=IF(ISTEXT(RC[-5]), COUNTA(OFFSET(FuncListStart,0,MATCH(RC[-5],
INDEX(PhaseRef, 0, 1), 0), 999, 1)), ““““),” _
            “=IF(AND(ISNUMBER(RC[-3]), ISNUMBER(R” & rPhaseRef.Row & “C[-
3])), RC[-3] / R” & rPhaseRef.Row & “C[-3], ““““),” _
            “=IF(AND(ISNUMBER(RC[-3]), ISNUMBER(R” & rPhaseRef.Row & “C[-
3])), RC[-3] / R” & rPhaseRef.Row & “C[-3], ““““)”)
         ‘.FormulaR1C1 = .Value
      End With
   Next iIndexP

   Call FormatTable(rPhaseRef, iNumDigits:=2)

   ‘ remove the color-fill that FormatTable puts for the columns we don’t want user-
editable
   With rPhaseRef.Offset(0, 3).Resize(columnsize:=5).Interior
      .Pattern = xlNone
      .TintAndShade = 0
      .PatternTintAndShade = 0
   End With

   With rPhaseRef.Offset(0, 1).Resize(rPhaseCount.Value, 1).Validation
      .Delete
      .Add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, Operator:=xlBetween,
Formula1:=“=INDEX(PLATFORMLIST,0,1)”
      .IgnoreBlank = True
      .InCellDropdown = True
      .ShowInput = True
      .ShowError = True
   End With

   ‘ $ Flt and $ Sim columns, formatted accounting-style
   rPhaseRef.Offset(0, 3).Resize(columnsize:=2).NumberFormat = FMT_ACCT
   rPhaseRef.Offset(0, 5).Resize(columnsize:=1).NumberFormat = “0”
```

```
‘ **********************************************************
‘ Pipeline Skills
‘ **********************************************************
‘ Function List
With rFuncListStart.CurrentRegion
   .ClearFormats
   .Value = .Value
End With

‘ Add a column before the extra column(s) so that it is not formatted in the table.
If iOldCount < rPhaseCount.Value Then
   ‘ insert new-old columns after old
   rFuncListStart.Offset(0, iOldCount + 1).Resize(columnsize:=rPhaseCount.Value -
iOldCount).EntireColumn.Insert shift:=xlToRight
ElseIf iOldCount > rPhaseCount.Value Then
   ‘ insert 1 row after new
   rFuncListStart.Offset(0,rPhaseCount.Value
1).Resize(columnsize:=1).EntireColumn.Insert shift:=xlToRight
End If

‘ Clear all pre-existing range names of “FuncList#” and redefine.
‘ (This ensures we don’t have unnecessary names floating around.)
For iIndex = ActiveWorkbook.Names.Count To 1 Step -1
   If Mid(ActiveWorkbook.Names(iIndex).Name, 1, 8) = “FuncList” And _
         ActiveWorkbook.Names(iIndex).Name <> “FuncListStart” Then
      ActiveWorkbook.Names(iIndex).Delete
   End If
Next iIndex

ReDim sPhases(1 To rPhaseCount.Value) As String
Dim sLineCount As String ‘ TODO
sLineCount = ““
For iIndexP = 1 To rPhaseCount.Value
   ActiveWorkbook.Names.Add _
      Name:=“FuncList” & iIndexP, _
      RefersTo:=“=OFFSET(Config!” & rFuncListStart.Offset(0, iIndexP).Address
& ,” 0, 0, COUNTA(Config!” & _
            rFuncListStart.Offset(0, iIndexP).EntireColumn.Address & “) - 1, 1)”
   sPhases(iIndexP) = “=INDEX(PhaseRef, “ & iIndexP & ,” 1)”
   sLineCount = sLineCount & ,” ISTEXT(RC[“ & iIndexP & “])”
Next iIndexP

‘ This will BREAK if this column mysteriously finds itself empty ...
‘ Well, not break, but it will fill this column in ALL ROWS IN THE WORKSHEET
‘ with this formula. Definitely not what we want/need, not certain if the
```

```
   ‘ computer will do it nicely and/or be able to recover. (In XL2010, that would
   ‘ fill 1,048,576 rows.)
   rFuncListStart.Offset(1, 0).Resize(rowsize:=CFG_FUNCLEN - 1).FormulaR1C1 = _
      “=IF(OR(“ & Mid(sLineCount, 2) & “), R[-1]C + 1, ““““)”

   With rFuncListStart.Offset(-1, 1).Resize(columnsize:=rPhaseCount.Value)
      .Value = sPhases
      .Formula = .Value
   End With

   Call FormatTable(rFuncListStart.Offset(0,1).Resize(CFG_FUNCLEN,
rPhaseCount.Value), iNumDigits:=2)

End Sub

‘ ***********************************************************
‘ MakeParametersPage()
‘ Prereq: Config page properly and completely filled out.
‘ Result: Deletes, recreates, and populates the Parameters page.
‘ Note: the following pages depend on this data and therefore are also deleted
‘       when the Parameters page is deleted:
‘       MasterDB, Download Page, {Primary,Intermediate,Advanced,FRS} Model
‘ ***********************************************************

Sub MakeParametersPage()
   Dim saveDisplayAlerts As Boolean
   Dim bPageExists As Boolean, rMBRet As VbMsgBoxResult
   Dim wsParms As Worksheet
   Dim rSkillsList As Range, rPhaseRef As Range
   Dim rFuncList() As Range
   Dim rFirewalls As Range
   Dim rDegrades() As Range
   Dim sPhase As String
   Dim iIndex As Integer, iIndex2 As Integer

   Application.ScreenUpdating = False
   saveDisplayAlerts = Application.DisplayAlerts

   Set rSkillsList = Range(“SkillsList”)
   Set rPhaseRef = Range(“PhaseRef”)

   ReDim rFuncList(rPhaseRef.Rows.Count) As Range
   ReDim rDegrades(rPhaseRef.Rows.Count) As Range

   For iIndex = 1 To rPhaseRef.Rows.Count
```

```
      Set rFuncList(iIndex) = Range(“FuncList” & iIndex)
   Next iIndex

   ‘ Check for the existence of all dependent worksheets.
   ‘ (If DEBUGFLAG, only check Parameters, all others ignore.)
   bPageExists = False
   bPageExists = bPageExists Or WorksheetExists(“Parameters”)
   If (Not DEBUGFLAG) Then
      bPageExists = bPageExists Or WorksheetExists(“MasterDB”)
      bPageExists = bPageExists Or WorksheetExists(“Download Page”)
      For iIndex = 1 To rPhaseRef.Rows.Count
         bPageExists = bPageExists Or WorksheetExists(“Model” & iIndex)
      Next iIndex
   End If

   ‘ Act on the existence of at least one worksheet ... the user isn’t notified of
   ‘ the specific worksheet, assuming they can figure it out.
   If bPageExists Then
      rMBRet = MsgBox(“Regenerating the Parameters worksheet will change results in
several other worksheets,” & _
                  “some of which are present. In order to continue, these worksheets will be
deleted.” & vbCrLf & _
                  “[ Parameters, MasterDB, Download Page, and Model* ]” & _
                  vbCrLf & vbCrLf & _
                  “Is that okay?” & _
                  vbCrLf & vbCrLf & _
                  “If not, click ““No”“ and back up the relevant tabs before trying this
again.,” _
                  Buttons:=vbYesNo Or vbQuestion Or vbDefaultButton2, Title:=“Delete
Existing Pages?”)
      ‘ vbYesNo gives the two buttons
      ‘ vbQuestion adds a question-mark icon (vbInformation is another)
      ‘ vbDefaultButton2 means default to the second button in case the user hits enter
right away
      ‘   (i.e., default to *not* changing things)

      If rMBRet = vbNo Then
         Exit Sub
      Else
         Application.DisplayAlerts = False
         If WorksheetExists(“Parameters”) Then
            Sheets(“Parameters”).Delete
         End If ‘ If WorksheetExists(“Parameters”) ...

         If (Not DEBUGFLAG) Then
```

```
            If WorksheetExists(“MasterDB”) Then
               Sheets(“MasterDB”).Delete
            End If ‘ If WorksheetExists(“MasterDB”) ...

            If WorksheetExists(“Download Page”) Then
               Sheets(“Download Page”).Delete
            End If ‘ If WorksheetExists(“Download Page”) ...

            For iIndex = 1 To rPhaseRef.Rows.Count
               If WorksheetExists(“Model” & iIndex) Then
                  Sheets(“Model” & iIndex).Delete
               End If ‘ If WorksheetExists(“Model” & iIndex) ...
            Next iIndex
         End If

         Application.DisplayAlerts = saveDisplayAlerts
      End If ‘ If rMBRet = vbNo ...
   End If ‘ If bPageExists ...

   Set wsParms = Sheets.Add(, after:=wsConfig)
   wsParms.Name = “Parameters”

   Set rFirewalls = wsParms.Range(STARTCELL_PARAMETERS).Resize(2 *
rPhaseRef.Rows.Count, rSkillsList.Rows.Count)
   rSkillsList.Resize(, 1).Copy
   rFirewalls.Offset(-1, 0).Resize(1).PasteSpecial xlPasteValues, , False, True
‘ SkipBlanks:=False, Transpose:=True

   For iIndex = 1 To rPhaseRef.Rows.Count
      rFirewalls.Offset(2 * (iIndex - 1), -1).Resize(2, 1) = _
            Application.Transpose(Array(rPhaseRef(iIndex, 1) & “ Enter,” _
                           rPhaseRef(iIndex, 1) & “ Exit”))
   Next iIndex

   rFirewalls.Name = “Firewalls”
   rFirewalls.ColumnWidth = 12.57     ‘ MAGIC NUMBER
   rFirewalls.Offset(0, -1).Resize(1, 1).EntireColumn.autofit

   Call FormatTable(rFirewalls, bDataEntry:=True, iNumDigits:=4)

   ‘ ********************************************************
   ‘ Set up each DEGRADE VALUES table

   For iIndex = rPhaseRef.Rows.Count To 1 Step -1
      If iIndex = rPhaseRef.Rows.Count Then
```

```
         Set rDegrades(iIndex) = rFirewalls.Offset(0, rSkillsList.Rows.Count +
1).Resize(rFuncList(iIndex).Rows.Count, 2 + 2 * rPhaseRef.Rows.Count)
      Else
         ‘ magic number: 5 = 1 + # of rows between each DEG block
         Set rDegrades(iIndex) = rDegrades(iIndex + 1).End(xlDown).Offset(5,
0).Resize(rFuncList(iIndex).Rows.Count, 2 + 2 * rPhaseRef.Rows.Count)
      End If

      rDegrades(iIndex).Name = “Degrades” & iIndex
      sPhase = rPhaseRef(iIndex, 1)

      With rDegrades(iIndex).Resize(1, 1)
         .Offset(-3, 2) = sPhase & “ MEDIA DEGRADES TO ...”

         For iIndex2 = rPhaseRef.Rows.Count To 1 Step -1
            .Offset(-2, 2 * (rPhaseRef.Rows.Count - iIndex2 + 1)) = .”.. “ &
rPhaseRef(iIndex2, 1)
            .Offset(-2, 2 * (rPhaseRef.Rows.Count - iIndex2 +
1)).Resize(columnsize:=2).Merge
            .Offset(-1, 2 * (rPhaseRef.Rows.Count - iIndex2 + 1)).Resize(1, 2) =
Array(“Flt,” “Sim”)
         Next iIndex2
         .Offset(-1, 0).Resize(1, 2) = Array(“#,” “Function”)
      End With
      rDegrades(iIndex).Resize(rFuncList(iIndex).Rows.Count, 1).Value =
Range(“FuncListStart”).Resize(rFuncList(iIndex).Rows.Count, 1).Value
      rDegrades(iIndex).Offset(0, 1).Resize(, 1).Value = rFuncList(iIndex).Value
      Call FormatTable(rDegrades(iIndex).Offset(0, 2).Resize(, 2 *
rPhaseRef.Rows.Count), bDataEntry:=True, iNumDigits:=2)
      rDegrades(iIndex).Offset(0, 2).NumberFormat = “0.00”

   Next iIndex

   ‘ ************************************************************
   ‘ Cleanup
   rDegrades(1).Resize(1, 2).EntireColumn.autofit
   rDegrades(1).Offset(0, 2).ColumnWidth = 8 ‘ magic number: 8, arbitrary

   ‘ ************************************************************
   ‘ Add the button
   Dim btnMasterDB As Button

   With Range(PR_BTN_MASTERDB)
      Set btnMasterDB = ActiveSheet.Buttons.Add(.Left, .Top, .Width, .Height)
   End With
   btnMasterDB.OnAction = “MakeMasterDBPage”
   btnMasterDB.Characters.Text = “Make Master DB Worksheet”

   With Range(PR_BTN_MASTERDB_HELP)
      .Value = “?”
      .HorizontalAlignment = xlCenter
      .AddComment (“Fill out the table of Enter/Exit table as well” & vbCrLf & _
               “the “ & rPhaseRef.Rows.Count & “ tables for media degrades.” & vbCrLf
& _
               vbCrLf & _
               “When complete, click this button to continue.”)
      .Comment.Shape.TextFrame.AutoSize = True
   End With

   ‘ *******************************************************
   ‘ Restore control to the user
   rFirewalls(1, 1).Select
   wsParms.Activate

   Application.CutCopyMode = False
   Application.DisplayAlerts = saveDisplayAlerts
   Application.ScreenUpdating = True
End Sub


‘ *******************************************************
‘ MakeMasterDBPage()
‘ Prereq: Config/Parameters page properly and completely filled out.
‘ Result: Deletes, recreates, and populates the MasterDB page, only
‘        including those column the user needs to edit; all other columns
‘        are added in MakeDownloadPage()
‘ Note: the following pages depend on this data and therefore are also deleted
‘      when the MasterDB page is deleted:
‘      Download Page, {Primary,Intermediate,Advanced,FRS} Model
‘ *******************************************************

Sub MakeMasterDBPage()
   Dim saveDisplayAlerts As Boolean, saveSelectionW As Worksheet, saveSelectionR
As Range

   Dim bPageExists As Boolean, rMBRet As VbMsgBoxResult
   Dim wsMasterDB As Worksheet
   Dim rSkillsList As Range, rPhaseRef As Range
   Dim iIndex As Integer, iIndexP As Integer, iIndexS As Integer
   Application.ScreenUpdating = False
   Set saveSelectionW = Selection.Worksheet
   Set saveSelectionR = Selection
   saveDisplayAlerts = Application.DisplayAlerts

   Set rSkillsList = Range(“SkillsList”)
   Set rPhaseRef = Range(“PhaseRef”)

   ‘ Check for the existence of all dependent worksheets.
   ‘ (If DEBUGFLAG, only check MasterDB, all others ignore.)
   bPageExists = False
   bPageExists = bPageExists Or WorksheetExists(“MasterDB”)
   If (Not DEBUGFLAG) Then
      bPageExists = bPageExists Or WorksheetExists(“Download Page”)
      For iIndex = 1 To rPhaseRef.Rows.Count
         bPageExists = bPageExists Or WorksheetExists(rPhaseRef(iIndex, PR_PHASE)
& “ Model”)
      Next iIndex
   End If

   ‘ Act on the existence of at least one worksheet ... the user isn’t notified of
   ‘ the specific worksheet, assuming they can figure it out.
   If bPageExists Then
      rMBRet = MsgBox(“Generating the MasterDB worksheet will change results in
several other worksheets,” & _
                  “some of which are present. In order to continue, these worksheets will be
deleted.” & vbCrLf & _
                  “[ MasterDB, Download Page, and all Model pages ]” & _
                  vbCrLf & vbCrLf & _
                  “Is that okay?” & _
                  vbCrLf & vbCrLf & _
                  “If not, click ““No”“ and back up the relevant tabs before trying this
again.,” _
                  Buttons:=vbYesNo Or vbQuestion Or vbDefaultButton2, Title:=“Delete
Existing Pages?”)
      ‘ vbYesNo gives the two buttons
      ‘ vbQuestion adds a question-mark icon (vbInformation is another)
      ‘ vbDefaultButton2 means default to the second button in case the user hits enter
right away
      ‘   (i.e., default to *not* changing things)

      If rMBRet = vbNo Then
         Exit Sub
      Else
         Application.DisplayAlerts = False
```

```
        If WorksheetExists(“MasterDB”) Then
          Sheets(“MasterDB”).Delete
        End If ‘ If WorksheetExists(“MasterDB”) ...

        If (Not DEBUGFLAG) Then
          If WorksheetExists(“Download Page”) Then
            Sheets(“Download Page”).Delete
          End If ‘ If WorksheetExists(“Download Page”) ...

          For iIndex = 1 To rPhaseRef.Rows.Count
            If WorksheetExists(“Model” & iIndex) Then
              Sheets(“Model” & iIndex).Delete
            End If ‘ If WorksheetExists(“Model” & iIndex) ...
          Next iIndex
        End If

        Application.DisplayAlerts = saveDisplayAlerts
      End If ‘ If rMBRet = vbNo ...
    End If ‘ If bPageExists ...

    Set wsMasterDB = Sheets.Add(, after:=Sheets(“Parameters”))
    wsMasterDB.Name = “MasterDB”

    With wsMasterDB.Range(STARTCELL_MASTERDB).Offset(0, -1)
      .Offset(, MDB_BLOCKS).Resize(1, MDB_NUMCOL).Value = _
          Array(“Blocks,” “Phase,” “Block ID,” “Title,” “Function,” “Medium,” “Flt or
Sim,” “Baseline Hours”)

      For iIndexS = 1 To rSkillsList.Rows.Count
        With .Offset(, MDB_SKILLS + iIndexS - 1)
          .Value = “Skill “ & iIndexS
          .AddComment Text:=rSkillsList(iIndexS, 1).Value
        End With
      Next iIndexS
    End With

    wsMasterDB.ListObjects.Add(xlSrcRange, _
      Range(STARTCELL_MASTERDB,
Range(STARTCELL_MASTERDB).CurrentRegion), , xlYes).Name = “MasterTable”

    rSkillsList.Resize(, 1).Copy
    wsMasterDB.Range(STARTCELL_MASTERDB).Offset(-1, MDB_SKILLS -
1).PasteSpecial xlPasteValues, , SkipBlanks:=False, Transpose:=True
```

```
   MsgBox “Please fill in this master database with appropriate data.,”
Buttons:=vbInformation
   ‘ vbInformation adds an info “i” icon (vbQuestion is another)

   ‘ *************************************************************
   ‘ Set up the continuation button
   Dim btnRemainder As Button

   With Range(MDB_BTN_DLPAGE)
      Set btnRemainder = ActiveSheet.Buttons.Add(.Left, .Top, .Width, .Height)
   End With
   btnRemainder.OnAction = “MakeRemainingPages”
   btnRemainder.Characters.Text = “Make Remaining Pages”

   With Range(MDB_BTN_DLPAGE_HELP)
      .Value = “?”
      .HorizontalAlignment = xlCenter
      .AddComment (“Fill out the Master DB.” & vbCrLf & _
               vbCrLf & _
               “When complete, click this button to make the remainder of the model.”)
      .Comment.Shape.TextFrame.AutoSize = True
   End With
   ‘ *************************************************************
   ‘ Restore control to the user.

   ‘ Two alternatives for handling the selection:
   ‘ (1) Ensure the previous selection is selected
   ‘ (2) Select the first cell for data entry in MasterDB
   wsMasterDB.Activate
   wsMasterDB.Range(STARTCELL_MASTERDB).Offset(1, 0).Select

   Application.CutCopyMode = False
   Application.DisplayAlerts = saveDisplayAlerts
   Application.ScreenUpdating = True
End Sub


‘ *************************************************************
‘ MakeRemainingPages()
‘ Prereq: Config, Parameters, and MasterDB pages properly and completely filled out.
‘ Result: Adds columns to the MasterDB page/range/table;
‘        Optionally hides new columns ;
‘        Optionally deletes and recreates the Download Page and all Model pages.
‘ Note: the following pages depend on this data and therefore are also deleted
‘      when the MasterDB page is modified:
```

```
‘      {Primary,Intermediate,Advanced,FRS} Model
‘ *******************************************************

Sub MakeRemainingPages()
   Dim saveDisplayAlerts As Boolean, saveSelectionW As Worksheet, saveSelectionR
As Range
   Dim bPageExists As Boolean, rMBRet As VbMsgBoxResult

   Dim wsMasterDB As Worksheet, wsDownloadPage As Worksheet
   Dim rNewColumns As Range
   Dim rSkillsList As Range, rPhaseRef As Range
   Dim sFormulaFlt As String, sFormulaSim As String, sFormulaMax As String
   Dim iIndex As Integer, iIndexS As Integer, iIndexP As Integer, iOffset As Integer

   Dim rFirewalls As Range
   Dim rDownloadTableRange As Range
   Dim iNumRows As Integer, iNumCols As Integer, sTmp As String
   Dim rTmp As Range ‘ for iterating over things ...
   Dim rDiffTable As Range, rBaselineTable As Range, rWhatIfTable As Range
   Dim rHoursComparison As Range, rAnnualCostDiff As Range

   Dim rCostData As Range, rHoursData As Range

   Application.ScreenUpdating = False
   saveDisplayAlerts = Application.DisplayAlerts

   Call StatusInitiate

   Set wsMasterDB = Sheets(“MasterDB”)

   Set rSkillsList = Range(“SkillsList”)
   Set rPhaseRef = Range(“PhaseRef”)
   Set rFirewalls = Range(“Firewalls”)

   Call StatusUpdate(“Checking for existing tables ...”)
   ‘ Check for the existence of all dependent worksheets.
   ‘ (If DEBUGFLAG, only check MasterDB, all others ignore.)
   bPageExists = False
   ‘ this checks if MasterDB has already been modified/refined/augmented
   bPageExists = bPageExists Or Not
wsMasterDB.ListObjects(1).HeaderRowRange.Find(“Cost”) Is Nothing
   bPageExists = bPageExists Or WorksheetExists(“Download Page”)
   If (Not DEBUGFLAG) Then
      For iIndexP = 1 To rPhaseRef.Rows.Count
         bPageExists = bPageExists Or WorksheetExists(“Model” & iIndexP)
```

```
        Next iIndexP
    End If
    ‘ Act on the existence of at least one worksheet ... the user isn’t notified of
    ‘ the specific worksheet, assuming they can figure it out.
    If bPageExists Then
        rMBRet = MsgBox(“Creating the Download Page worksheet will change results in
several other worksheets,” & _
                    “some of which are present. In order to continue, these worksheets will be
deleted.” & vbCrLf & _
                    “[ (changes to) MasterDB, Download Page, and all Model pages ]” & _
                    vbCrLf & vbCrLf & _
                    “Is that okay?” & _
                    vbCrLf & vbCrLf & _
                    “If not, click ““No”“ and back up the relevant tabs before trying this
again.,” _
                    Buttons:=vbYesNo Or vbQuestion Or vbDefaultButton2, Title:=“Delete
Existing Pages?”)
        ‘ perhaps this will make the window go away
        Application.ScreenUpdating = True
        Application.ScreenUpdating = False
        ‘ vbYesNo gives the two buttons
        ‘ vbQuestion adds a question-mark icon (vbInformation is another)
        ‘ vbDefaultButton2 means default to the second button in case the user hits enter
right away
        ‘    (i.e., default to *not* changing things)

        If rMBRet = vbNo Then
            Exit Sub
        Else
            Application.DisplayAlerts = False

            ‘ if “Cost” exists, remove it and everything to the right in this table
            On Error GoTo Skip:
                Dim iRow As Integer
                iRow = Application.Match(“Cost,” Range(“MasterDB!MasterTable”).Rows(0),
0)
                Range(Range(“MasterDB!MasterTable”).Offset(-1, iRow - 1).Resize(1, 1), _
                    Range(“MasterDB!MasterTable”).Offset(-1, iRow - 1).Resize(1,
1).End(xlToRight)).EntireColumn.Delete
Skip:
            On Error GoTo 0

            If WorksheetExists(“Download Page”) Then
                Sheets(“Download Page”).Delete
            End If ‘ If WorksheetExists(“Download Page”) ...
        If (Not DEBUGFLAG) Then
           For iIndexP = 1 To rPhaseRef.Rows.Count
              If WorksheetExists(“Model” & iIndexP) Then
                 Sheets(“Model” & iIndexP).Delete
              End If ‘ If WorksheetExists(“Model” & iIndexP) ...
           Next iIndexP
        End If

        Application.DisplayAlerts = saveDisplayAlerts
     End If ‘ If rMBRet = vbNo ...
  End If ‘ If bPageExists ...

  ‘ ***********************************************************
  ‘ *** Download Page, part 1
  ‘ ***********************************************************
  Call StatusUpdate(“Creating the Download Page ...”)

  Dim sTable As String
  sTable = “Download Page”

  Set wsDownloadPage = Sheets.Add(after:=Sheets(“MasterDB”))
  wsDownloadPage.Name = “Download Page”
  iNumRows = Range(“MasterTable[[#All]]”).Rows.Count - 1
  iNumCols = 3 + 2 * rPhaseRef.Rows.Count + 4 ‘ magic numbers: 3 is the number of
pre-phase columns,
                                    ‘ and 4 is the number of post-phase columns only
Download Page

  ‘ change DL_HOURDIFF if more columns added or shifted
  Set rDownloadTableRange =
wsDownloadPage.Range(STARTCELL_DLPAGE).Offset(1,
0).Resize(Range(“MasterTable”).Rows.Count, iNumCols)

  Dim iCol As Integer
  iCol = 0

  Dim aStr() As String
  ReDim aStr(1 To iNumCols) As String

  With rDownloadTableRange
     Call StatusUpdate

     ‘.Name = “DownloadTableRange”
```

```
        aStr(DL_BLOCKS) = “Blocks”
        aStr(DL_PHASE) = “Phase”
        aStr(DL_BLOCKID) = “Block ID”

        For iIndexP = 1 To rPhaseRef.Rows.Count
            aStr(DL_PHASES + iIndexP - 1) = “Phase “ & iIndexP
            aStr(rPhaseRef.Rows.Count + DL_PHASES + iIndexP - 1) = “OrigPhase “ &
iIndexP
        Next iIndexP
        iCol = 3 + 2 * rPhaseRef.Rows.Count ‘ magic number: 3 is the number of pre-phase
columns

        aStr(iCol + DL_BASELINEHRS) = “Baseline Hours”
        aStr(iCol + DL_TITLE) = “Title”
        aStr(iCol + DL_COSTDIFF) = “Cost Difference”
        aStr(iCol + DL_HOURDIFF) = “Hour Difference”
        ‘ ASSERT: we have filled all 15 of aStr (based on 4 phases)

        .Offset(-1, 0).Resize(rowsize:=1).Value = aStr
        wsDownloadPage.ListObjects.Add(xlSrcRange, .CurrentRegion, , xlYes).Name =
“DownloadTable”

        Range(“DownloadTable[Blocks]”).Resize(Range(“MasterTable”).Rows.Count,
1).Formula = “=MasterTable[@Blocks]”
        Range(“DownloadTable[Blocks]”).Value =
Range(“DownloadTable[Blocks]”).Value
        Range(“DownloadTable[Phase]”).Value = Range(“MasterTable[Phase]”).Value
        Range(“DownloadTable[Block ID]”).Value = Range(“MasterTable[Block
ID]”).Value

        For iIndexP = 1 To rPhaseRef.Rows.Count
            Range(“DownloadTable[[Phase “ & iIndexP & “]]”).Formula = “=( [@Phase] =
““““ & rPhaseRef(iIndexP, 1) & ““““ ) * MasterTable[[Baseline Hours]]”

            Range(“DownloadTable[[OrigPhase “ & iIndexP & “]]”).Value =
Range(“DownloadTable[[Phase “ & iIndexP & “]]”).Value
            Range(“DownloadTable[[#Headers],[Phase “ & iIndexP & “]]”).NumberFormat =
““““““ & rPhaseRef(iIndexP, 1) & ““““  @”
        Next iIndexP

        Range(“DownloadTable[Baseline Hours]”).Value = Range(“MasterTable[Baseline
Hours]”).Value
        Range(“DownloadTable[Title]”).Value = Range(“MasterTable[Title]”).Value

        Call StatusUpdate
        Dim sTemp As String ‘ TODO
        For iIndex = 1 To iNumRows
            Call StatusUpdate
            .Cells(iIndex, DL_BLOCKID).AddComment (.Cells(iIndex, iCol +
DL_TITLE).Value)
            .Cells(iIndex, DL_BLOCKID).Comment.Shape.TextFrame.AutoSize = True
        Next iIndex

    End With

    Range(“DownloadTable”).Columns.autofit

    sTemp = Range(“DownloadTable[Phase]”).Cells(1, 1).Value
    Range(“DownloadTable[Phase]”).Cells(1, 1).Value = “W” ‘ W == wide character
    Range(“DownloadTable[Phase]”).Cells(1, 1).Columns.autofit
    Range(“DownloadTable[Phase]”).Cells(1, 1).Value = sTemp

    With Range(“DownloadTable[[Phase 1]:[Phase “ & rPhaseRef.Rows.Count & “]]”)
        .FormatConditions.Add Type:=xlExpression, Formula1:=“=NOT(“ & _
            Range(“DownloadTable[Phase 1]”).Resize(1, 1).Address(rowabsolute:=False,
columnabsolute:=False) & “ = “ & _
            Range(“DownloadTable[OrigPhase 1]”).Resize(1,
1).Address(rowabsolute:=False, columnabsolute:=False) & “)”
        With .FormatConditions(.FormatConditions.Count)
            .StopIfTrue = False
            .Interior.PatternColorIndex = xlAutomatic
            .Interior.Color = 65535
            .Interior.TintAndShade = 0
        End With

        .FormatConditions.Add Type:=xlExpression, Formula1:=“=NOT(SUM(“ & _
            Range(“DownloadTable[[Phase 1]:[Phase “ & rPhaseRef.Rows.Count &
“]]”).Resize(rowsize:=1).Address(rowabsolute:=False, columnabsolute:=True) & “) =
“ & _
            Range(“DownloadTable[Baseline
Hours]”).Resize(rowsize:=1).Address(rowabsolute:=False, columnabsolute:=True) & “)”
        With .FormatConditions(.FormatConditions.Count)
            .StopIfTrue = False
            .Font.Color = -16776961
            .Font.TintAndShade = 0
        End With
    End With

    ‘ ***********************************************************
```

```
   ‘ *** Model pages
   ‘ ***********************************************************

   For iIndexP = 1 To rPhaseRef.Rows.Count
      Call StatusUpdate(“Creating the “ & rPhaseRef(iIndexP, 1) & “ Model ...”)
      Call MakeModelPage(iIndexP)
   Next iIndexP

   ‘ ***********************************************************
   ‘ *** Download Page, part 2
   ‘ ***********************************************************

   Call StatusUpdate(“Finishing the Download Page ...”)

   Set rDiffTable = Range(“DownloadTable”).Offset(1,
Range(“DownloadTable”).Columns.Count + 3).Resize(2 * rPhaseRef.Rows.Count,
rSkillsList.Rows.Count)

   Const DL_TABLE_SEP As Integer = 3 ‘ header row plus two gap rows
   Set rBaselineTable = rDiffTable.Offset(DL_TABLE_SEP + 2 *
rPhaseRef.Rows.Count, 0)
   Set rWhatIfTable = rBaselineTable.Offset(DL_TABLE_SEP + 2 *
rPhaseRef.Rows.Count, 0)

   ‘ We’ll set up rDiffTable last since it depends on the other two ...
   ‘ perhaps not strictly necessary, but it makes sense.

   With rBaselineTable
      .Name = “BaselineTable”
      .Offset(-1, -1).Resize(1, 1).Value = “Baseline”
      .Offset(-1, 0).Resize(rowsize:=1).Value = Range(“Firewalls”).Offset(-1,
0).Resize(rowsize:=1).Value
      .Offset(0, -1).Resize(columnsize:=1).Value = Range(“Firewalls”).Offset(0, -
1).Resize(columnsize:=1).Value
      .Offset(0, -1).Resize(rFirewalls.Rows.Count, 1).Value =
Range(“Firewalls”).Offset(0, -1).Resize(rFirewalls.Rows.Count, 1).Value
      .Offset(-1, 0).Resize(1, rFirewalls.Columns.Count).Value =
Range(“Firewalls”).Offset(-1, 0).Resize(1, rFirewalls.Columns.Count).Value
   End With

   With rWhatIfTable
      .Name = “WhatIfTable”
      .Offset(-1, -1).Resize(.Rows.Count + 1, .Columns.Count + 1).Value = _
         rBaselineTable.Offset(-1, -1).Resize(.Rows.Count + 1, .Columns.Count +
1).Value
```

```
        .Offset(-1, -1).Resize(1, 1).Value = “What-If”
    End With

    With rDiffTable
        .Name = “DiffTable”
        .Offset(-1, -1).Resize(.Rows.Count + 1, .Columns.Count + 1).Value = _
            rBaselineTable.Offset(-1, -1).Resize(.Rows.Count + 1, .Columns.Count +
1).Value
        .Offset(-1, -1).Resize(1, 1).Value = “Difference”
    End With

    Call StatusUpdate

    For iIndexP = 1 To rPhaseRef.Rows.Count
        Range(“DownloadTable[Phase “ & iIndexP & “]”).Value = Range(“Table1[[Phase
“ & iIndexP & “]]”).Value

        For iIndexS = 1 To rSkillsList.Rows.Count
            Call StatusUpdate
            rBaselineTable(2 * iIndexP - 1, iIndexS).Formula = _
                “=INDEX(Table” & iIndexP & “[Baseline GALE “ & iIndexS & “], 1)”
            rBaselineTable(2 * iIndexP, iIndexS).Formula = _
                “=INDEX(Table” & iIndexP & “[Baseline GALE “ & iIndexS & “], “ & _
                “COUNT(Table” & iIndexP & “[Baseline GALE “ & iIndexS & “]))”

            rWhatIfTable(2 * iIndexP - 1, iIndexS).Formula = _
                “=INDEX(Table” & iIndexP & “[What If GALE “ & iIndexS & “], 1)”
            rWhatIfTable(2 * iIndexP, iIndexS).Formula = _
                “=INDEX(Table” & iIndexP & “[What If GALE “ & iIndexS & “], “ & _
                “COUNT(Table” & iIndexP & “[What If GALE “ & iIndexS & “]))”

            rDiffTable(2 * iIndexP - 1, iIndexS).Formula = _
                “=“ & rWhatIfTable(2 * iIndexP - 1, iIndexS).Address & “ - “ &
rBaselineTable(2 * iIndexP - 1, iIndexS).Address
            rDiffTable(2 * iIndexP, iIndexS).Formula = _
                “=“ & rWhatIfTable(2 * iIndexP, iIndexS).Address & “ - “ & rBaselineTable(2
* iIndexP, iIndexS).Address
        Next iIndexS
    Next iIndexP

    rDiffTable.Offset(0, -1).Resize(1, rSkillsList.Rows.Count + 1).EntireColumn.autofit

    Call StatusUpdate
    Call FormatTable(rBaselineTable, bDataEntry:=False)
    Call StatusUpdate
```

```
   Call FormatTable(rWhatIfTable, bDataEntry:=False)
   Call StatusUpdate
   Call FormatTable(rDiffTable, bDataEntry:=False)

   Set rCostData = rDiffTable.Offset(2 - rDiffTable.Cells(1, 1).Row,
rSkillsList.Rows.Count + 2).Resize(rPhaseRef.Rows.Count, 3) ‘ magic numbers
   Set rHoursData = rCostData.Offset(0, 5) ‘ initially set one more away so FormatTable
can work cleanly, then we’ll delete the empty column
   rCostData.Name = “CostData”
   rHoursData.Name = “HoursData”

   With rCostData.Resize(1, 1)
      .Offset(-1, 0) = “Baseline”
      .Offset(-1, 1) = “What If”
      .Offset(-1, 2) = “Difference”
      .Offset(rPhaseRef.Rows.Count, 0) = “Cost”
      .Offset(rPhaseRef.Rows.Count, 2).FormulaR1C1 = “=SUM(R[-4]C:R[-1]C)”
   End With

   With rHoursData.Offset(-1, -1).Resize(1, 1)
      .Value = “$1,000”
      .HorizontalAlignment = xlCenter
      .Font.Bold = False
   End With
   rHoursData.Offset(0, -1).Resize(rPhaseRef.Rows.Count, 1).Value =
rPhaseRef.Resize(rPhaseRef.Rows.Count, 1).Value
   rHoursData.Offset(-1, 0).Resize(1, 3).Value = rCostData.Offset(-1, 0).Resize(1,
3).Value
   rHoursData.Offset(rPhaseRef.Rows.Count, 0).Resize(1, 1) = “Hours”
   rHoursData.Offset(rPhaseRef.Rows.Count, 2).Resize(1, 1).FormulaR1C1 =
“=SUM(R[-4]C:R[-1]C)”

   For iIndexP = 1 To rPhaseRef.Rows.Count
      rCostData(iIndexP, 1).Formula = “=SUM(Table” & iIndexP & “[Baseline
Cost])/1000”
      rCostData(iIndexP, 2).Formula = “=SUM(Table” & iIndexP & “[What If
Cost])/1000”
      rCostData(iIndexP, 3).FormulaR1C1 = “=rc[-1] - rc[-2]”

      rHoursData(iIndexP, 1).Formula = “=SUM(DownloadTable[OrigPhase “ & iIndexP
& “])”
      rHoursData(iIndexP, 2).Formula = “=SUM(DownloadTable[Phase “ & iIndexP &
“])”
      rHoursData(iIndexP, 3).FormulaR1C1 = “=rc[-1] - rc[-2]”
   Next iIndexP
   Call StatusUpdate
   Call FormatTable(rCostData, bDataEntry:=False)
   Call StatusUpdate
   Call FormatTable(rHoursData, bDataEntry:=False, iNumDigits:=1)
   rHoursData.Offset(rPhaseRef.Rows.Count, 2).Resize(1, 1).NumberFormat = “0.0”

   rHoursData.Offset(0, -2).Resize(1, 1).EntireColumn.Delete ‘ this was an empty
column kept in for ease of FormatTable

   Union(rCostData.Resize(4, 4), rHoursData).EntireColumn.autofit


   ‘ *** summary table, next to hourdata
   Set rHoursComparison = rHoursData.Offset(0, rHoursData.Columns.Count +
2).Resize(2, rSkillsList.Rows.Count)

   rHoursComparison.Offset(-1, 0).Resize(rowsize:=1).Value =
Range(“Firewalls”).Offset(-1, 0).Resize(1, rFirewalls.Columns.Count).Value
   rHoursComparison.Offset(0, -1).Resize(columnsize:=1).Value =
WorksheetFunction.Transpose(Array(“Baseline,” “What If”))
   rHoursComparison.Rows(1).FormulaArray = “=“ & rBaselineTable.Rows(2 *
rPhaseRef.Rows.Count).Address
   rHoursComparison.Rows(2).FormulaArray = “=“ & rWhatIfTable.Rows(2 *
rPhaseRef.Rows.Count).Address

   rHoursComparison.Name = “HoursComparison”

   Call FormatTable(rHoursComparison, bDataEntry:=False, iNumDigits:=2)

   For iIndexS = 1 To rSkillsList.Rows.Count
      With rHoursComparison(2, iIndexS)
         .FormatConditions.Add Type:=xlCellValue, Operator:=xlLess, _
            Formula1:=“=index(SkillsList,” & iIndexS & ,”2)”
         .FormatConditions(.FormatConditions.Count).SetFirstPriority

         With .FormatConditions(1).Interior
            .PatternColorIndex = xlAutomatic
            .Color = 65535
            .TintAndShade = 0
         End With
         .FormatConditions(1).StopIfTrue = False
      End With
   Next iIndexS
   ‘ *** Annual cost difference
   Set rAnnualCostDiff = rHoursComparison.Offset(3, 1).Resize(1, 2)
   With rAnnualCostDiff
      .Resize(1, 1).Formula = “=AnnualThroughput * “ &
rCostData.Offset(rPhaseRef.Rows.Count, 2).Resize(1, 1).Address
      .Offset(0, -1).Resize(1, 1).Value = “Annual Cost Difference”
      .Merge
   End With
   Call FormatTable(rAnnualCostDiff, bDataEntry:=False, rowHeaderAlign:=xlRight)

   rAnnualCostDiff.Name = “AnnualCostDiff”

   ‘ Accounting format, so that 0 looks like “$-,” etc.
   Union(rCostData, _
       rCostData.Offset(rPhaseRef.Rows.Count, 2).Resize(1, 1), _
       rAnnualCostDiff).NumberFormat = FMT_ACCT

   ‘ *******************************************************
   ‘ Baseline GRAPH
   wsDownloadPage.Activate

   Dim chBaseline As Chart, chWhatIf As Chart, iTmp As Integer
   Set chBaseline = wsDownloadPage.ChartObjects.Add(DL_CHART_LEFT,
DL_CHART_TOP, DL_CHART_WIDTH, DL_CHART_HEIGHT).Chart
   Set chWhatIf = wsDownloadPage.ChartObjects.Add(DL_CHART_LEFT,
DL_CHART_TOP + DL_CHART_HEIGHT + 20, DL_CHART_WIDTH,
DL_CHART_HEIGHT).Chart

   With chBaseline
      .ChartType = xl3DColumn
      .SetSourceData Source:=rBaselineTable.CurrentRegion
      .Location where:=xlLocationAsObject, Name:=“Download Page”
      .HasTitle = True
      .ChartTitle.Text = “Baseline”
   End With

   With chWhatIf
      .ChartType = xl3DColumn
      .SetSourceData Source:=rWhatIfTable.CurrentRegion
      .Location where:=xlLocationAsObject, Name:=“Download Page”
      .HasTitle = True
      .ChartTitle.Text = “What If”
   End With

   For iTmp = 1 To wsDownloadPage.ChartObjects.Count
```

```
    With wsDownloadPage.ChartObjects(iTmp).Chart
        .Legend.Delete
        .PlotVisibleOnly = False
        .Axes(xlValue).TickLabels.NumberFormat = “0.00”
        .Axes(xlValue).MajorGridlines.Select
        .SeriesCollection(2 * rPhaseRef.Rows.Count).Select
        .SeriesCollection(2 * rPhaseRef.Rows.Count).ApplyDataLabels
        .SeriesCollection(2 * rPhaseRef.Rows.Count).DataLabels.NumberFormat =
“#,##0.00”
    End With
  Next iTmp

  ‘ ********************************************************
  ‘ Pretty things up

  Range(Range(“DownloadTable[[#Headers],[OrigPhase 1]]”).EntireColumn, _
      rCostData.Resize(1, 1).Offset(0, -1).EntireColumn).Hidden = True
  wsDownloadPage.Cells(Range(“DownloadTable”).Resize(1, 1).Row, _
                rCostData.Resize(1, 1).Column).Select
  ActiveWindow.freezepanes = True
  Range(“DownloadTable[[Phase 1]]”).Cells(1, 1).Select

  ‘ ********************************************************
  ‘ Add the buttons
  Call StatusReset

  Dim btnReset As Button, btnSave As Button ‘ TODO

  With Range(DL_BTN_RESET)
    Set btnReset = ActiveSheet.Buttons.Add(.Left, .Top, .Width, .Height)
  End With
  btnReset.OnAction = “DownloadReset”
  btnReset.Characters.Text = “Reset Hours”

  With Range(DL_BTN_RESET_HELP)
    .Value = “?”
    .HorizontalAlignment = xlCenter
        .AddComment (“Resets all ““What If”“ hours for all phases to ““Baseline
        Hours”.”“ & vbCrLf & vbCrLf &”NOTE! This action cannot be undone! If you
        want to save the current state” & vbCrLf & “then click ““Save Scenario”“ first.”)
    .Comment.Shape.TextFrame.AutoSize = True
  End With

  Call StatusReset
```

```
   With Range(DL_BTN_SAVE)
      Set btnSave = ActiveSheet.Buttons.Add(.Left, .Top, .Width, .Height)
   End With
   btnSave.OnAction = “DownloadSave”
   btnSave.Characters.Text = “Save Scenario”

   With Range(DL_BTN_SAVE_HELP)
      .Value = “?”
      .HorizontalAlignment = xlCenter
.AddComment (“Saves this scenario work-in-progress as a new worksheet.” & vbCrLf &
vbCrLf &”NOTE! All numbers and charts on the new worksheet will be static” & vbCrLf
& “and will not reflect changes in any other portion of this spreadsheet.”)
      .Comment.Shape.TextFrame.AutoSize = True
   End With

   ‘ **********************************************************
   ‘ Restore control to the user.
   Call StatusReset
   Application.CutCopyMode = False
   Application.DisplayAlerts = saveDisplayAlerts
   Application.ScreenUpdating = True
End Sub
‘ **********************************************************
‘ MakeModelPage(PhaseString)
‘ Input: integer, referring to the numbered phase to build;
‘       1-based, meaning “1” refers to Primary in the example usage,
‘       as defined in PhaseRef, first column
‘ Output: deletes Model# worksheet if it exists,
‘        copies from previous Model sheet if it exists, MasterDB otherwise,
‘        Adds remaining columns and formulas
‘ Assumptions:
‘   - StatusBar work is being done and StatusInitiate has been run
‘   - Application.ScreenUpdating (and similar visuals-controllers) have been set
‘ **********************************************************
Sub MakeModelPage(ByVal iPhase As Integer)
   Dim bCopyMaster As Boolean
   Dim wsModel As Worksheet
   Dim rPhaseRef As Range, rSkillsList As Range, sTable As String
   Dim iIndexS As Integer, iIndexP As Integer
   Dim aHeaders() As String, aFormulas() As String
   Dim sDegrades As String
   Dim iLastCol As Integer ‘ used to append to the DataTable
   Dim sPhase As String
   Dim sTemp As String
   Set rPhaseRef = Range(“PhaseRef”)
   Set rSkillsList = Range(“SkillsList”)


   sPhase = rPhaseRef(iPhase, 1)
   If WorksheetExists(“Model” & iPhase) Then
      Application.DisplayAlerts = False
      Sheets(“Model” & iPhase).Delete
   End If ‘ if worksheetexists(“Model” & iphase) ...

   Call StatusUpdate

   If WorksheetExists(“Model” & (iPhase - 1)) Then
      bCopyMaster = False
      ‘ previous model exists, copying it
      Sheets(“Model” & (iPhase - 1)).Copy after:=Sheets(“Model” & (iPhase - 1))
   Else
      bCopyMaster = True
      ‘ first time or previous model does not exist, copying MasterDB
      Sheets(“MasterDB”).Copy after:=Sheets(“MasterDB”)
   End If ‘ If WorksheetExists(“Model” & (iPhase - 1)) ...

   Set wsModel = ActiveSheet

   Call StatusUpdate

   With wsModel
      .Name = “Model” & iPhase
      .ListObjects(1).Unlist ‘ remove the datatable
      .ListObjects.Add(xlSrcRange, _
         Intersect(Range(STARTCELL_MASTERDB).CurrentRegion,
Range(STARTCELL_MASTERDB).CurrentRegion.Offset(1, 0)), , xlYes).Name =
“Table” & iPhase
   End With

   sTable = “Table” & iPhase

   ReDim aFormulas(5 + 3 * rPhaseRef.Rows.Count) As String ‘ magic number: 5
includes model, BL min, BL max, WI hours, and HourDiff
   ReDim aHeaders(5 + 3 * rPhaseRef.Rows.Count) As String  ‘... and 3* is because we
have 3 columns per phase: ph, phmin, and platform

   iLastCol = Range(“Table” & iPhase).Columns.Count

   If bCopyMaster Then
```

```
    For iIndexP = 1 To rPhaseRef.Rows.Count
        Call StatusUpdate

        sDegrades = sDegrades & ,” Degrades” & iIndexP
    Next iIndexP
    sDegrades = Mid(sDegrades, 3)

    ‘ starting “from scratch,” need to build in all new columns
    ‘ the initial block of columns, static, throw it at the end of the table
    Range(sTable & “[#Headers]”).Offset(0, Range(sTable &
“[#Headers]”).Columns.Count).Resize(1, columnsize:=7).Value = Array( _
        “Model,” _
        “Baseline Min,” _
        “Baseline Max,” _
        “What If Hours,” _
        “Hour Difference,” _
        “Baseline Cost,” _
        “What If Cost”)

    For iIndexP = 1 To rPhaseRef.Rows.Count
        Call StatusUpdate
        ‘ added per-phase
        Range(sTable & “[#Headers]”).Offset(0, Range(sTable &
“[#Headers]”).Columns.Count).Resize(1, columnsize:=3).Value = Array( _
            “Phase “ & iIndexP, _
            “Phase Min “ & iIndexP, _
            “Platform “ & iIndexP)
    Next iIndexP

    For iIndexS = 1 To rSkillsList.Rows.Count
        Call StatusUpdate
        ‘ added per-skill
        ‘ magic number: 6 is the number of columns we’re adding here
        Range(sTable & “[[#Headers],[Skill “ & iIndexS & “]]”).Offset(0, 1).Resize(1,
6).EntireColumn.Insert
        Range(sTable & “[[#Headers],[Skill “ & iIndexS & “]]”).Offset(0, 1).Resize(1,
6).Value = Array( _
                “Degr Skill “ & iIndexS, _
                “Baseline % Total “ & iIndexS, _
                “Baseline GALE “ & iIndexS, _
                “What If % Total “ & iIndexS, _
                “What If GALE “ & iIndexS, _
                “Difference “ & iIndexS)
    Next iIndexS
```

```
      For iIndexP = 1 To rPhaseRef.Rows.Count
         Call StatusUpdate
         ‘ added per-phase
         Range(sTable & “[Phase “ & iIndexP & “]”).Resize(1, columnsize:=3).Value =
Array( _
            “=([@Phase] = ““““ & rPhaseRef(iIndexP, 1) & ““““) * [@[Baseline Hours]],” _
            “=([@Phase] = ““““ & rPhaseRef(iIndexP, 1) & ““““) * [@[Baseline Min]],” _
               “=VLOOKUP([@Function], CHOOSE( MATCH([@Phase],
INDEX(PhaseRef,,1), 0), “ & sDegrades & “), “ & _
            “IF([@[Flt or Sim]]=““Flt”,” 0, 1) + “ & 2 * (rPhaseRef.Rows.Count + 1) - 2 *
iIndexP + 1 & ,” FALSE)”)
         Range(sTable & “[Platform “ & iIndexP & “]”).Resize(1, 1) = 0

         With Range(sTable & “[[#Headers],[Phase “ & iIndexP & “]]”).Offset(-1, 0)
            .Resize(1, 1) = rPhaseRef(iIndexP, 1)
            .Resize(1, 3).Merge
         End With
      Next iIndexP
   End If ‘ If bCopyMaster ...

   ‘ initial block
   ‘ Model, Baseline Min, Baseline Max, What If Hours, Hour Difference, Baseline Cost,
What If Cost
   Range(sTable & “[Model]”).Resize(1, columnsize:=7).Value = Array( _
      “,” _
      “=LOOKUP([@[Baseline Hours]], {0, 0.5, 1}),” _
      “=VLOOKUP([@Phase], PhaseRef, 3, FALSE) * [@[Baseline Hours]],” _
      “=DownloadTable[@[Phase “ & iPhase & “]],” _
      “=[@[What If Hours]] - [@[Phase “ & iPhase & “]],” _
      “=([@Phase] = ““““ & sPhase & ““““) * IF([Flt or Sim] = ““Flt”,” INDEX(PhaseRef,
“ & iPhase & ,” 4), INDEX(PhaseRef, “ & iPhase & ,” 5)) * [@[Baseline Hours]],” _
      “=IF([Flt or Sim] = ““Flt”,” INDEX(PhaseRef, “ & iPhase & ,” 4),
INDEX(PhaseRef, “ & iPhase & ,” 5)) * [@[What If Hours]]”)
   Range(sTable & “[Model]”).Resize(1, columnsize:=5).Formula = Range(sTable &
“[Model]”).Resize(1, columnsize:=5).Value

   For iIndexS = 1 To rSkillsList.Rows.Count
      Call StatusUpdate
      ‘ added per-skill
      ‘ leave out “Difference #” since it’s being corrupted as we mod row 1, so we’ll add
it later
      Range(sTable & “[Degr Skill “ & iIndexS & “]”).Resize(1,
columnsize:=5).FormulaR1C1 = Array( _
         “=[@[Platform “ & iPhase & “]] * [@[Skill “ & iIndexS & “]],” _
         “=[@[Phase “ & iPhase & “]] * [@[Degr Skill “ & iIndexS & “]] / “ & _
```

```
            "SUMPRODUCT([Phase " & iPhase & "], [Degr Skill " & iIndexS & "]),” _
         "=( INDEX(Firewalls, " & (2 * iPhase) & ," " & iIndexS & ") -
INDEX(Firewalls, " & (2 * iPhase - 1) & ," " & iIndexS & ") ) * " & _
            "[@[Baseline % Total " & iIndexS & "]] + R[-1]C," _
         "=([@[What If Hours]] * [@[Degr Skill " & iIndexS & "]]) / " & _
            "SUMPRODUCT([Phase " & iPhase & "], [Degr Skill " & iIndexS & "]),” _
         "=( INDEX(Firewalls, " & (2 * iPhase) & ," " & iIndexS & ") -
INDEX(Firewalls, " & (2 * iPhase - 1) & ," " & iIndexS & ") ) * " & _
            "[@[What If % Total " & iIndexS & "]] + R[-1]C")
      Range(sTable & "[Degr Skill " & iIndexS & "]").Resize(1,
columnsize:=6).FormulaR1C1 = Range(sTable & "[Degr Skill " & iIndexS &
"]").Resize(columnsize:=6).Value

      Application.AutoCorrect.AutoFillFormulasInLists = False
      Range(sTable & "[Baseline GALE " & iIndexS & "]").Resize(1, 1).Formula =
"=INDEX(Firewalls, " & (2 * iPhase - 1) & ," " & iIndexS & ")"
      Union( _
         Range(sTable & "[Baseline GALE " & iIndexS & "]").Resize(1, 1), _
         Range(sTable & "[What If GALE " & iIndexS & "]").Resize(1, 1)).Formula =
"=INDEX(Firewalls, " & (2 * iPhase - 1) & ," " & iIndexS & ")"
      If iPhase > 1 Then
         Range(sTable & "[What If GALE " & iIndexS & "]").Resize(1, 1).Formula = _
            Range(sTable & "[What If GALE " & iIndexS & "]").Resize(1, 1).Formula &
" + " & _
            "INDEX(Table" & (iPhase - 1) & "[Difference " & iIndexS & "], count(Table"
& (iPhase - 1) & "[Difference " & iIndexS & "]))"
      End If
      Application.AutoCorrect.AutoFillFormulasInLists = True

      Range(sTable & "[Difference " & iIndexS & "]").FormulaR1C1 = "=RC[-1] - RC[-
3]"

      Range(sTable & "[[#Headers],[Skill " & iIndexS & "]]").Offset(-1, 0).Resize(1,
7).Merge   ' magic number: 7 is the number columns per Skill
   Next iIndexS

   Range(sTable).EntireColumn.Hidden = True

   Dim rRng As Range
   Set rRng = Range("A1") ' random
   For iIndexS = 1 To rSkillsList.Rows.Count
      Set rRng = Union(rRng, _
                  Range(sTable & "[Baseline GALE " & iIndexS & "]"), _
                  Range(sTable & "[What If GALE " & iIndexS & "]"))
   Next iIndexS
   rRng.NumberFormat = “0.0000”
   Set rRng = Union(rRng, Range(sTable & “[Blocks]”))
   With rRng.EntireColumn
      .Hidden = False
      .autofit
   End With

   Range(sTable).NumberFormat = “0.0000”
End Sub

‘ *********************************************************
‘ DownloadReset()
‘ Input: none
‘ Output: the columns DownloadTable[Phase #] are reset to their original values.
‘ *********************************************************
Sub DownloadReset()
   ‘ This banks on the fact that the Phase columns are collocated and
   ‘ the OrigPhase columns are immediately following them.
   Range(“DownloadTable[Phase
1]”).Resize(columnsize:=Range(“PhaseRef”).Rows.Count).Value = _
      Range(“DownloadTable[OrigPhase
1]”).Resize(columnsize:=Range(“PhaseRef”).Rows.Count).Value
End Sub

‘ *********************************************************
‘ DownloadSave()
‘ Input: none
‘ Output: the “Download Page” worksheet is cloned and all formulas
‘        within it are converted to values only (i.e., static).
‘        Subsequent saves will rename the new worksheet numerically
‘        such that scenarios are not overwritten.
‘ *********************************************************
Sub DownloadSave()
   Dim wsNew As Worksheet
   Dim iCount As Integer, iTmp As Integer

   Application.ScreenUpdating = False

   With ActiveWorkbook
      .Sheets(“Download Page”).Copy after:=.Sheets(“Download Page”)
      Set wsNew = .Sheets(.Sheets(“Download Page”).Index + 1)
   End With

   iCount = 1
   For iTmp = 1 To ActiveWorkbook.Sheets.Count
```

```
        If Mid(ActiveWorkbook.Sheets(iTmp).Name, 1, 9) = “Scenario “ Then
            ‘ this is certainly not a flawless method ...
            iCount = WorksheetFunction.Max(iCount, 1 +
Int(Mid(ActiveWorkbook.Sheets(iTmp).Name, 10)))
        End If
    Next iTmp
    wsNew.Name = “Scenario “ & iCount

    If wsNew.Names.Count > 0 Then
        For iTmp = wsNew.Names.Count To 1 Step -1 ‘ backwards since deleting changes
all indexes
            Range(wsNew.Names(iTmp)).Value = Range(wsNew.Names(iTmp)).Value
        Next iTmp
    End If ‘ If wsNew.Names.Count > 0 ...

    If wsNew.Buttons.Count > 0 Then
        For iTmp = wsNew.Buttons.Count To 1 Step -1 ‘ backwards since deleting changes
all indexes
            wsNew.Buttons(iTmp).Delete
        Next iTmp
    End If
    With Union(wsNew.Range(DL_BTN_SAVE_HELP),
wsNew.Range(DL_BTN_RESET_HELP))
        .ClearContents
        .ClearComments
        .Clear
    End With

    With wsNew.Range(DL_BTN_SAVE).Resize(1, 1)
        .Value = “Scenario saved:”
        .Resize(1, 3).Merge
        .Offset(1, 0).Value = Date
        .Offset(1, 0).Resize(1, 3).Merge
        .Offset(2, 0).Value = Time
        .Offset(2, 0).Resize(1, 3).Merge
    End With

    Sheets(“Download Page”).Activate

    Application.ScreenUpdating = True

End Sub

‘ ************************************************************
‘ FormatTable()
‘ Input: a range to be formatted; just the data, the headers are found by expanding
‘       the selection.
‘ Result: Formatted as a table.
‘ Arguments:
‘   rRange     := range object to format, not including headers which will
‘              be found with .CurrentRegion
‘   bDataEntry := if True, this range is meant for user data entry and
‘              formatted with a yellow background; if False, no color.
‘              (Default: True)
‘   iNumDigits := the number of decimal digits to include, default = 4.
‘   *Align     := { xlLeft, xlCenter, xlRight }
‘ An initial “CurrentRegion” is taken of the range, causing it to expand to include
‘ all non-empty cells in a square, out to a “square moat” of empty cells. All
‘ cells are initially formatted like a header, and then the original range is
‘ properly formatted as data.
‘ ***********************************************************

Sub FormatTable(ByVal rRange As Range, Optional ByVal bDataEntry As Boolean =
True, _
            Optional ByVal iNumDigits As Integer = 4, Optional ByVal rowHeaderAlign
As Integer = xlCenter, _
            Optional ByVal columnHeaderAlign As Integer = xlCenter, Optional ByVal
cellAlign As Integer = xlCenter, _
            Optional ByVal ExpandRange As Boolean = True)
   Dim rTmp As Range

   If ExpandRange Then
      Set rTmp = rRange.CurrentRegion
   Else
      Set rTmp = rRange
   End If

   With rTmp.Font
      .Bold = True
      .Name = “Calibri”
      .Size = 12
      .Strikethrough = False
      .Superscript = False
      .Subscript = False
      .OutlineFont = False
      .Shadow = False
      .Underline = xlUnderlineStyleNone
      .ThemeColor = xlThemeColorLight1
      .TintAndShade = 0
      .ThemeFont = xlThemeFontMinor
```

```
End With

rRange.NumberFormat = “0.” & String(iNumDigits, “0”)
rRange.HorizontalAlignment = cellAlign
rRange.Offset(-1, 0).Resize(rowsize:=1).HorizontalAlignment = columnHeaderAlign
rRange.Offset(0, -1).Resize(columnsize:=1).HorizontalAlignment = rowHeaderAlign

With rRange.Font
   .Bold = False
End With

With rRange.Interior
   If bDataEntry Then
      .Pattern = xlSolid
      .PatternColorIndex = xlAutomatic
      .Color = 10092543
      .TintAndShade = 0
      .PatternTintAndShade = 0
   Else
      .Pattern = xlSolid
      .PatternColorIndex = xlAutomatic
      .ThemeColor = xlThemeColorDark1
      .TintAndShade = -0.149998474074526
      .PatternTintAndShade = 0
   End If
End With

Dim myBorders() As Variant, item As Variant
myBorders = Array(xlEdgeLeft, _
             xlEdgeTop, _
             xlEdgeBottom, _
             xlEdgeRight)
For Each item In myBorders
   With rRange.Borders(item)
      .LineStyle = xlContinuous
      .ColorIndex = xlAutomatic
      .TintAndShade = 0
      .Weight = xlMedium
   End With
Next item

myBorders = Array(xlInsideVertical, xlInsideHorizontal)
For Each item In myBorders
   With rRange.Borders(item)
      .LineStyle = xlContinuous
```

```
            .ColorIndex = xlAutomatic
            .TintAndShade = 0
            .Weight = xlHairline
        End With
    Next item
End Sub


‘ *********************************************************
‘ WorksheetExists
‘ Input: the name of a worksheet
‘ Output: True if a sheet with that name exists, False otherwise
‘ *********************************************************

Public Function WorksheetExists(ByVal WorksheetName As String) As Boolean
    On Error Resume Next
    WorksheetExists = (Sheets(WorksheetName).Name <> ““)
    On Error GoTo 0
End Function

‘ *********************************************************
‘ TableExists
‘ Input: the name of a (data) table, (previously known as an AutoFilter table)
‘ Output: True if a table with that name exists, False otherwise
‘ *********************************************************
Public Function TableExists(ByVal TableName As String) As Boolean
    On Error Resume Next
    TableExists = (ActiveSheet.ListObject(TableName).Name <> ““)
    On Error GoTo 0
End Function
‘ *********************************************************
‘ RangeNameExists
‘ Input: string, the name of a range
‘ Output: True if a range with that name exists locally or globally, false otherwise
‘ *********************************************************
Public Function RangeNameExists(ByVal RangeName As String) As Boolean
    On Error Resume Next
    RangeNameExists = False
    RangeNameExists = Len(Range(RangeName).Address) <> 0
End Function
‘ *********************************************************
‘ StatusInitiate
‘ Call this in the beginning of any function that will be using the status bar
‘ to ensure that the global variable is properly defined. (This would not be
‘ necessary if VBA allowed Public Const Array.)
‘ *************************************************************
Sub StatusInitiate()
   SB_Message = ““‘
   SB_Index = 0
   If Not IsArray(SB_HeartBeat) Then
      SB_HeartBeat = Array(“|o--------|,” “|-o-------|,” “|--o------|,” “|---o-----|,” “|----o----
|,” “|-----o---|,” “|------o--|,” “|-------o-|,” _
                     “|--------o|,” “|-------o-|,” “|------o--|,” “|-----o---|,” “|----o----|,” “|---o-----
|,” “|--o------|,” “|-o-------|”)
      ‘‘‘ Alternatives:
      ‘ Array(.”,” “o,” “O,” “o,” .”,” “ “)
      ‘ Array(“\,” “|,” “/,” “-”)
   End If
End Sub
‘ *************************************************************
‘ StatusUpdate
‘ If called with a message, update to that message. With no arguments,
‘ the previous message is retained and that “heartbeat” is incremented.
‘ SB_HeartBeat should be an array that incrementally shows some form of
‘ progress. For aesthetics, it is helpful to keep all strings within the
‘ array of equal size; keep in mind that the font is a variable-width font,
‘ so a string of three spaces and a string of three M’s are different
‘ sizes.
‘ *************************************************************
Sub StatusUpdate(Optional ByVal sMessage As Variant)
   SB_Index = (SB_Index + 1) Mod (UBound(SB_HeartBeat) + 1)
   If Not IsMissing(sMessage) Then
      SB_Message = sMessage
   End If
   Application.StatusBar = “Progress:” & SB_HeartBeat(SB_Index) & “   “ &
SB_Message
   DoEvents
End Sub

‘ *************************************************************
‘ StatusReset
‘ I’m not certain this is necessary, but it clears the statusbar which
‘ eventually shows “Ready.”
‘ *************************************************************
Sub StatusReset()
   Application.StatusBar = False
End Sub

Sub StatusBarTest()
   Dim iCounter As Integer, iIndex As Integer, sMessages
```

```
    On Error GoTo ErrHandler:
    Call StatusInitiate
    Application.ScreenUpdating = False
    sMessages = Array("Refining MasterDB ...," "Starting Download Page ...," _
                "Making Primary Model ...," "Making Intermediate Model ...," _
                "Making Advanced Model ...," "Making FRS Model ...," _
                "Finishing Download Page ...")
    For iCounter = 0 To 200
        If (iCounter Mod 200 / (UBound(sMessages) + 1)) = 0 Then
            Call StatusUpdate(sMessages(Int(iCounter / (200 / (UBound(sMessages) + 1)))))
        Else
            Call StatusUpdate
        End If
        Sleep (40)
    Next iCounter
ErrHandler:
    On Error GoTo 0
    Call StatusReset
    Application.ScreenUpdating = True
End Sub

' ************************************************************
' AddrRow(string)
' Input: an address string in A1 format, e.g. "C4" or "AQ92"
' Output: only the row (numerical) component, e.g. "4" or "92"
' Note: input can be either relative ("C4") or absolute ("$C$4"), though
'       output will always output relative-only
' ************************************************************
Function AddrRow(sRange As String)
    With CreateObject("vbscript.regexp")
        .Global = False
        .Pattern = "([0–9]+)"
        If .test(sRange) Then
            AddrRow = .Execute(sRange)(0)
        Else
            AddrRow = sRange
        End If
    End With
End Function

' ************************************************************
' AddrCol(string)
' Input: an address string in A1 format, e.g. "C4" or "AQ92"
' Output: only the column (alphabetic) component, e.g. "C" or "AQ"
' Note: input can be either relative ("C4") or absolute ("$C$4"), though
```

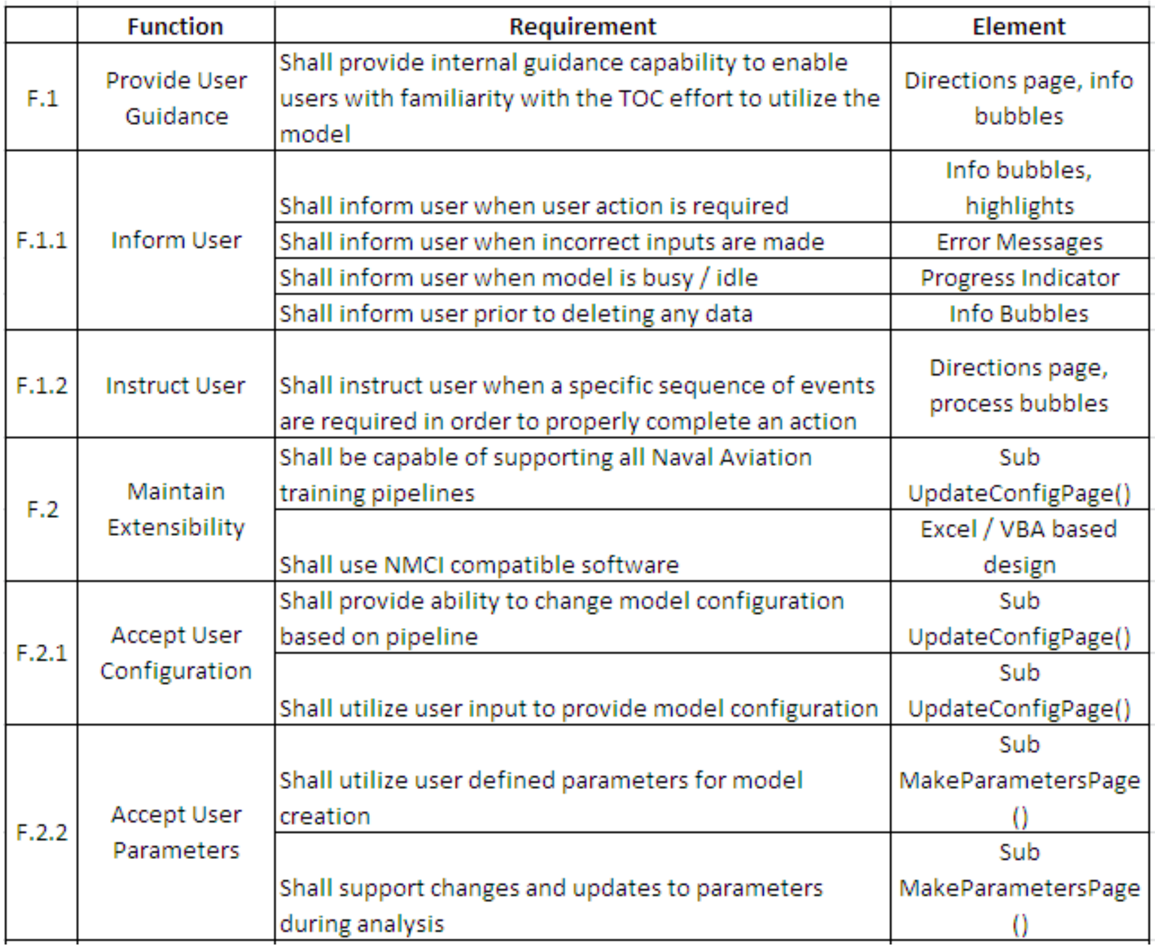
```
‘      output will always output relative-only
‘ ********************************************************
Function AddrCol(sRange As String)
   With CreateObject(“vbscript.regexp”)
      .ignorecase = True
      .Global = False
      .Pattern = “([A-Z]+)”
      If .test(sRange) Then
         AddrCol = .Execute(sRange)(0)
      Else
         AddrCol = sRange
      End If
   End With
End Function

‘ ********************************************************
‘ AddrAbs(string)
‘ Input: an address string in A1 format, e.g. “C4” or “AQ92”
‘ Output: an absolute version of it, e.g. “$C$4” or “$AQ$92”
‘ Note: input can be either absolute or relative.
‘ ********************************************************
Function AddrAbs(sRange As String)
   AddrAbs = Range(sRange).Address(rowabsolute:=True, columnabsolute:=True)
End Function

‘ ********************************************************
‘ AddrRel(string)
‘ Input: an address string in A1 format, e.g. “C$4” or “$AQ$92”
‘ Output: a relative version of it, e.g. “C4” or “AQ92”
‘ Note: input can be either absolute or relative.
‘ ********************************************************
Function AddrRel(sRange As String)
   AddrAbs = Range(sRange).Address(rowabsolute:=False, columnabsolute:=False)
End Function
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F: REQUIREMENT / ELEMENT MAPPING



	Function	Requirement	Element
F.1	Provide User Guidance	Shall provide internal guidance capability to enable users with familiarity with the TOC effort to utilize the model	Directions page, info bubbles
F.1.1	Inform User	Shall inform user when user action is required	Info bubbles, highlights
		Shall inform user when incorrect inputs are made	Error Messages
		Shall inform user when model is busy / idle	Progress Indicator
		Shall inform user prior to deleting any data	Info Bubbles
F.1.2	Instruct User	Shall instruct user when a specific sequence of events are required in order to properly complete an action	Directions page, process bubbles
F.2	Maintain Extensibility	Shall be capable of supporting all Naval Aviation training pipelines	Sub UpdateConfigPage()
		Shall use NMCI compatible software	Excel / VBA based design
F.2.1	Accept User Configuration	Shall provide ability to change model configuration based on pipeline	Sub UpdateConfigPage()
		Shall utilize user input to provide model configuration	Sub UpdateConfigPage()
F.2.2	Accept User Parameters	Shall utilize user defined parameters for model creation	Sub MakeParametersPage ()
		Shall support changes and updates to parameters during analysis	Sub MakeParametersPage ()

Table 4. NAPA Model Elements Mapped to Requirements (Part 1)

	Function	Requirement	Element
F.3	Manipulate Data	Shall support basic forms of query either with resident Excel functionality or with VBA functionality	Excel Table functionality
F.3.1	Accept Data Input	Shall accept user-defined data sets in an identified common format	Excel Table Format / Common data columns
F.3.2	Store Data	Shall provide the ability to store data for future manipulation and analysis	Save / Save As functionality
F.3.3	Process Data	Shall utilize Excel and VBA for data processing	Excel / VBA based design
		Shall provide the ability to search and sort data according to user defined parameters	Excel Table Sort Functionality
F.4	Provide Analysis	Shall support NAWCTSD HP/ISP GALE methodology	Sub MakeRemainingPages ()
		Shall provide timely analysis results in a user-friendly format	Download Page
		Shall provide cost and proficiency variations from baseline	Download Page
		Shall inform user of breeched thresholds	Download Page / Conditional formatting
F.4.1	Compare Scenarios	Be operable by an individual with a baseline of knowledge in Naval Aviation training	Design concept
		Shall provide the ability to save multiple scenarios for comparison	Scenario save button
F.4.2	Support Optimization	Shall provide ability to search for optimal training scenarios given desired cost or proficiency values as targets	Framework provided No Optimization Capability
F.5	Be Supportable	Be maintainable by an individual with above average computing skills (programming)	Commented Code
		Shall be executable on any Navy Marine Corps Intranet (NMCI) computer	Excel / VBA based design

Table 5. NAPA Model Elements Mapped to Requirements (Part 2)

LIST OF REFERENCES

Commander Naval Air Forces. 2012. “Naval Aviation Mission.”. Accessed December 15, 2012. http://www.cnaf.navy.mil/nae/.

Chief of Naval Air Training. “CNATRA: Training Future Aerial Warriors.” Accessed October 12, 2012. http://www.cnatra.navy.mil.

Blanchard, Benjamin S., and Wolter J Fabrycky. 2011, *Systems Engineering and Analysis.* (5th ed.). Upper Saddle River: Prentice Hall.

Bovey, Rob, Dennis Wallentin, Steven Bullen, and John Green. 2009. *Professional Excel Development: The Definitive Guide to Developing Applications Using Microsoft Excel, VBA, and .NET.* Boston: Pearson Education, 2009.

Langford, Gary O. 2012, *Engineering Systems Integration: Theory, Metrics, and Methods.*(1st ed.). Boca Raton: CRC Press.

Walkenbach, John. 2010. *Excel 2010: Power Programming with VBA.* Indianapolis , IN: Wiley Publishing.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California